MAISON MELOTTI

CREAMA: UMA ARQUITETURA DE REFERÊNCIA PARA O DESENVOLVIMENTO DE SISTEMAS COLABORATIVOS MÓVEIS BASEADOS EM COMPONENTES

VITÓRIA

2014

MAISON MELOTTI

CREAMA: UMA ARQUITETURA DE REFERÊNCIA PARA O DESENVOLVIMENTO DE SISTEMAS COLABORATIVOS MÓVEIS BASEADOS EM COMPONENTES

Dissertação apresentada ao Curso de Ciência da Computação, Departamento de Informática, Universidade Federal do Espírito Santo, como requisito parcial para a obtenção do título de Mestre em Informática.

Orientadora: Profa. Dra. Roberta Lima Gomes
Co-orientador: Prof. Dr. Marco Aurélio Gerosa

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

CENTRO TECNOLÓGICO

DEPARTAMENTO DE INFORMÁTICA

VITÓRIA

2014

Dados Internacionais de Catalogação-na-publicação (CIP)
(Biblioteca Setorial Tecnológica,
Universidade Federal do Espírito Santo, ES, Brasil)

Melotti, Maison, 1990-

M528c CReAMA: uma arquitetura de referência para o desenvolvimento de sistemas colaborativos móveis baseados em componentes / Maison Melotti. – 2014.

94 f. : il.

Orientador: Roberta Lima Gomes.

Coorientador: Marco Aurélio Gerosa.

Dissertação (Mestrado em Informática) – Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Groupware. 2. Computação móvel. 3. Componente de software. I. Gomes, Roberta Lima. II. Gerosa, Marco Aurélio. III. Universidade Federal do Espírito Santo. Centro Tecnológico. IV. Título.

CDU: 004

Dissertação de Mestrado sob o título “CReAMA: Uma Arquitetura de Referência para o Desenvolvimento de Sistemas Colaborativos Móveis Baseados em Componentes”, defendida por Maison Melotti e aprovada em 29 de Agosto de 2014, em Vitória, Estado do Espírito Santo, pela banca examinadora constituída pelos professores:

_____

Profa. Dra. Roberta Lima Gomes
Universidade Federal do Espírito Santo
Orientadora

_____

Prof. Dr. Marco Aurélio Gerosa
Universidade de São Paulo
Co-orientador

_____

Prof. Dr. Vitor Estêvão Silva Souza
Universidade Federal do Espírito Santo

_____

Prof. Dr. Mateus Conrad Barcellos da Costa
Instituto Federal do Espírito Santo

À Pâmella,
pela força e compreensão e
meus pais e família,
pelo carinho e apoio.

AGRADECIMENTOS

Holy Roberta.

Holy Renilda.

Holy Jota.

Holy Pâmella.

Holy Google.

Agradeço de todo o coração pela paciência, apoio e sabedoria que me foi passada ao longo dessa tortuosa estrada que foi o Mestrado. Eu vou levar tudo isso comigo para outros lugares e pessoas, afinal, a estrada é a vida. Muito obrigado por tudo.

“Acreditai! E vivereis para sempre – Acreditai que vivestes eternamente – deixai para trás as fortalezas e penitências do sofrimento solitário na escuridão da terra, pois há mais coisas na vida além da terra, há Luz por Toda Parte, vede -”

Jack Kerouac.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

RESUMO

CREAMA: UMA ARQUITETURA DE REFERÊNCIA PARA O DESENVOLVIMENTO DE SISTEMAS COLABORATIVOS MÓVEIS BASEADOS EM COMPONENTES

Sistemas colaborativos são sistemas que suportam dois ou mais usuários engajados em uma tarefa comum, fornecendo uma interface para um ambiente compartilhado para esses usuários e as aplicações móveis se tornaram uma parte importante das ferramentas de suporte à colaboração. Quando usados no contexto de mobilidade, sistemas colaborativos podem ser chamados de sistemas colaborativos móveis. Esses sistemas possibilitam, por exemplo, que equipes cooperem enquanto estão em movimento, aumentando o potencial da colaboração.

Ferramentas para apoiar desenvolvedores de software são muito importantes de uma forma geral, mas no que diz respeito à colaboração móvel, ferramentas se tornam fundamentais principalmente por se tratar de uma área multidisciplinar que lida com muitas questões técnicas. Porém, sem uma base de requisitos ou modelos que mapeiam esses requisitos para se apoiar, muitas vezes o ferramental pode se tornar inutilizado ou pouco útil. Torna-se com isso de grande importância prover suporte ao desenvolvimento de sistemas colaborativos móveis por meio de um conjunto de ferramentas e métodos que apoiem o desenvolvedor na criação desses sistemas.

Devido a isto, neste trabalho foi desenvolvida CReAMA, uma arquitetura de referência para orientar o desenvolvimento de sistemas colaborativos móveis orientados a componentes. Além da definição da arquitetura de referência também foi realizada uma avaliação de CReAMA com o intuito de observar com uma visão abrangente a validade da proposta.

Palavras-Chave: Groupware, Computação Móvel, Componente de Software.

ABSTRACT

CREAMA: A REFERENCE ARCHITECTURE FOR DEVELOPMENT OF COMPONENT-BASED MOBILE GROUPWARE SYSTEMS

Collaborative systems are systems that support two or more users engaged in a common task, providing an interface to a shared environment for these users and mobile applications have become an important part of the collaboration support tools. When used in the context of mobility, collaborative systems can be called mobile collaborative systems. These systems allow, for example, that cooperate teams while in motion, increasing the potential of collaboration.
Tools to support software developers are very important in general, but with regard to mobile collaboration tools become paramount, especially because it is a multidisciplinary field that deals with many technical issues. However, without a foundation of requirements or models that map these requirements to support, often the tooling may become unusable or less useful. Therefore, it is very important to provide support to the development of mobile collaborative systems through a set of tools and methods to support the developer in creating these systems.
Because of this, in this work was developed CReAMA, a reference architecture to guide the development of component-based mobile collaborative systems. In the reference architecture definition was also performed a CReAMA evaluation in order to comply with a comprehensive view the validity of the proposal.

Key words: Groupware, Mobile Computing, Software Component.

1 INTRODUÇÃO

1 INTRODUÇÃO

Sistemas colaborativos são sistemas que suportam dois ou mais usuários engajados em uma tarefa comum, fornecendo uma interface para um ambiente compartilhado para esses usuários. Sistemas colaborativos se distinguem dos sistemas tradicionais distribuídos primeiramente pela necessidade dos usuários em alcançar uma meta em comum (IVAN e CIUREA, 2009).

Em um sistema colaborativo, o controle é realizado sobre a colaboração entre usuários e com isso: (i) o controle da concorrência é mais complexo devido ao à usuários compartilhando recursos; (ii) em muitos sistemas há a necessidade de suporte à comunicação entre os usuários; e (iii) geralmente há a necessidade de implementação de políticas distribuídas que permitam que os usuários se coordenem enquanto acessam os recursos compartilhados (GOMES et al., 2011).

É importante ressaltar que o desenvolvimento de sistemas colaborativos é intrinsecamente complexo. Sistemas colaborativos são tipicamente difíceis de serem projetados e implementados visto que apresentam os desafios da área de sistemas distribuídos somados aos problemas de sistemas multiusuários (GOMES et al., 2011). De acordo com Gerosa e Fuks (2008), a área de sistemas colaborativos é altamente interdisciplinar e os processos de negócio que definem a dinâmica do grupo de trabalho são difíceis de modelar e apoiar.

Para Fonseca et al. (2009), as aplicações móveis se tornaram uma parte importante das ferramentas de suporte à colaboração. De acordo com Papadopoulos (2006), esses sistemas possibilitam, por exemplo, que equipes cooperem enquanto estão em movimento, aumentando o potencial da colaboração.

As oportunidades de colaborar são ampliadas pela computação móvel (FILIPPO et al., 2011). Mas essas oportunidades também acarretam desafios adicionais para o desenvolvimento de sistemas colaborativos móveis, especialmente em termos de compartilhamento e sincronização de informação por meio de dispositivos móveis. Um problema refere-se à duração do período em que um usuário móvel permanece conectado. Essa duração depende de vários fatores como tempo de vida limitado da bateria, largura de banda instável das redes sem fio ou usuários em locomoção (o que pode causar frequentes interrupções do canal de

comunicação). Para Le e Nygard (2007), tudo isso pode resultar em longos atrasos ou longos períodos de interrupção para as atividades colaborativas de usuários de dispositivos móveis. Além disso, a limitação nos recursos computacionais de dispositivos móveis pode atrapalhar operações já iniciadas, devido, por exemplo, à necessidade de mais dados que ainda não foram totalmente transferidos para o dispositivo (LE e NYGARD, 2007).

Novos paradigmas de colaboração precisam ser desenvolvidos para levar em conta fatores como a disponibilidade do usuário, a variabilidade em recursos dos dispositivos e a conectividade de rede não confiável em ambientes de computação móvel (LITIU e ZEITOUN, 2004). Características como essas acarretam muitos desafios estudados em trabalhos relacionados à colaboração móvel, especialmente em termos de compartilhamento de informação por meio desses dispositivos.

Para Herskovic et al. (2011), as dificuldades inerentes à mobilidade são causadas, em parte, por requisitos mal elicitados e falta de experiência de programadores em relação a esse tipo de sistema. Processos de colaboração móvel não têm sido estudados extensivamente e é difícil para desenvolvedores de software analisarem e modelarem esse tipo de trabalho. Métodos de análise de software tradicionais não consideram vários aspectos importantes de colaboração móvel (HERSKOVIC et al., 2011). De acordo com Herskovic et al. (2009), questões como a mobilidade dos colaboradores, as diversas tecnologias disponíveis para apoiar a mobilidade (Wi-fi, WiMax, Bluetooth, etc.) e a mudança contínua nos cenários de colaboração trazem novos desafios para o desenvolvimento desses sistemas.

Torna-se com isso de grande importância prover suporte ao desenvolvimento de sistemas colaborativos móveis por meio de um conjunto de ferramentas e métodos que apoiem o desenvolvedor na criação desses sistemas. No contexto de sistemas colaborativos em geral, diferentes soluções são encontradas. Por exemplo, para o desenvolvimento de aplicações colaborativas na Web 2.0, foi proposto o *Groupware Workbench* (GW) (GROUPWARE WORKBENCH, 2013), que fornece um ferramental baseado em componentes para encapsular as complexidades técnicas envolvidas no desenvolvimento de sistemas colaborativos. Porém, vale ressaltar que no contexto de sistemas colaborativos móveis verifica-se uma carência de arquiteturas ou plataformas para auxiliar seus desenvolvedores. São encontrados na literatura apenas alguns trabalhos que propõem o uso de

ferramentas para o desenvolvimento de sistemas colaborativos móveis, porém com algumas limitações, como apresentado por Byrne (2011) e Bendel e Schuster (2012).

1.1 OBJETIVOS

Este trabalho tem como objetivo geral definir uma arquitetura de referência – denominada CReAMA (***C****omponent-Based* ***Re****ference* ***A****rchitecture for Collaborative* ***M****obile* ***A****pplications*) – para orientar o desenvolvimento de aplicações colaborativas para dispositivos móveis Android[1] baseadas em componentes. Neste trabalho, uma arquitetura de referência representa uma forma de se apresentar um padrão genérico para um projeto (ZAMBIASI, 2012). Com base nessa arquitetura, o desenvolvedor projeta, desenvolve e configura uma aplicação prototipando-a por meio de componentes reutilizáveis.

O desenvolvimento orientado a componentes apresenta algumas vantagens importantes no que diz respeito ao desenvolvimento de software, como melhor suporte à prototipação e evolução do software, melhor capacidade de adaptação, substituição dinâmica de partes do sistema, gerenciamento de mudanças, interoperabilidade, entre outras vantagens. Também optou-se neste trabalho por definir a arquitetura CReAMA com base na plataforma Android. O objetivo foi tornar a arquitetura de referência otimizada para uma determinada plataforma. A plataforma escolhida foi o Android visto que se trata de uma plataforma código aberto que vem sendo adotada por uma quantidade crescente de usuários e desenvolvedores em todo o mundo (VISION MOBILE, 2013).

O objetivo geral deste trabalho se compõe pelos seguintes objetivos específicos:

- Levantamento de requisitos gerais ou frequentes de sistemas colaborativos e requisitos de sistemas colaborativos móveis encontrados na literatura;

[1] http://developer.android.com/index.html

- Levantamento de requisitos de sistemas orientados a componentes para auxiliar na definição de requisitos de aplicações móveis baseadas em componentes;
- Definição e implementação de uma arquitetura de referência, com base nos requisitos estudados, para orientar o desenvolvimento de aplicações colaborativas para dispositivos móveis Android baseadas em componentes;
- Implementação de protótipos de diferentes sistemas colaborativos móveis para mostrar o uso da arquitetura proposta.

1.2 METODOLOGIA

Com o intuito de alcançar os objetivos mencionados na seção anterior, foi realizada uma pesquisa sobre técnicas atuais de desenvolvimento de sistemas colaborativos e sistemas colaborativos móveis. Essa pesquisa teve como objetivo identificar os principais estudos relacionados ao estado da arte do suporte metodológico e suporte computacional para o desenvolvimento desses sistemas.

Essa pesquisa também auxiliou no levantamento de requisitos para sistemas colaborativos móveis por meio de revisão bibliográfica. Depois do levantamento inicial dos requisitos, foi feito um mapeamento dos requisitos genéricos e frequentes encontrados na literatura. Foram inclusos alguns requisitos de sistemas groupware (como requisitos de comunicação) e de aplicações móveis (como requisitos de interface com o usuário).

Depois do mapeamento de requisitos, foi definida uma arquitetura de referência para orientar o desenvolvimento de aplicações colaborativas para dispositivos móveis Android baseadas em componentes com base nos requisitos identificados na etapa anterior. Em seguida foi realizada a implementação da arquitetura proposta, incluindo o modelo de componentes que pode ser utilizado, por exemplo, para agrupar e compor componentes.

Como prova de conceito, neste trabalho foram desenvolvidos protótipos de sistemas colaborativos móveis baseados na CReAMA. Dois protótipos foram clientes móveis de sistemas criados com base na extensão da bancada de componentes do Groupware-Workbench (GW). O GW fornece um ferramental baseado em componentes para encapsular as complexidades técnicas envolvidas no

desenvolvimento de sistemas colaborativos. Além disso, foram desenvolvidas duas versões de um sistema colaborativo fora do escopo do GW, denominado ConecteIdeias[2], uma versão baseada em CReAMA e outra versão sem a utilização da arquitetura de referência. Visou-se com isso fazer uma comparação de dificuldade de desenvolvimento de cada versão para ilustrar as vantagens e desvantagens da arquitetura definida neste trabalho.

1.3 CONTRIBUIÇÕES DO TRABALHO

As contribuições do trabalho são:

- Mapeamento de diferentes requisitos genéricos ou frequentes de sistemas colaborativos móveis encontrados na literatura. São inclusos requisitos de sistemas groupware e de aplicações móveis.
- Elaboração de um cenário motivador para auxiliar na definição de requisitos de aplicações móveis baseadas em componentes. O cenário ajudou a reforçar requisitos que já são encontrados na literatura de desenvolvimento de software baseado em componentes, desde que não foi encontrado nenhum cenário parecido na literatura.
- Definição de uma arquitetura de referência para orientar o desenvolvimento de aplicações colaborativas para dispositivos móveis Android baseadas em componentes. Na definição está incluso o modelo de componentes utilizado, além da definição da infraestrutura de execução desses componentes e uso de sensores por meio de componentes.
- Implementação da arquitetura proposta, incluindo o modelo de componentes que pode ser utilizado para agrupar/compor componentes gráficos, além da infraestrutura (junto com os sensores) que é utilizada pelos componentes para a utilização do serviço de execução de requisições, controle de concorrência, entre outras funcionalidades.

[2] http://conecteideias.com/

- Desenvolvimento de uma extensão do Groupware Workbench, denominada GWAndroid, por meio da implementação de um conjunto de componentes que podem ser utilizados no desenvolvimento de aplicações clientes móveis. Esses componentes incluem a versão para ambiente móvel de componentes já existentes para Web, além de novos componentes que não existiam anteriormente no GW, que possuem aspectos relacionados à mobilidade.
- Desenvolvimento de protótipos de aplicações clientes móveis de sistemas colaborativos construídos usando o GW: (i) um sistema de perguntas e respostas colaborativo, em que o usuário utiliza a aplicação para colaborar na definição de respostas sobre um determinado domínio; (ii) a versão móvel da rede social Arquigrafia – o projeto da rede social Arquigrafia visa prover para usuários interessados em arquitetura, um ambiente colaborativo para a visualização, interação e compartilhamento de imagens digitais de arquitetura na internet.
- Desenvolvimento de protótipos de aplicações clientes móveis do sistema ConecteIdeias, construídos com e sem o auxílio de CReAMA, juntamente com uma comparação da dificuldade de desenvolvimento de cada versão (utilizando e não utilizando CReAMA), com o intuito de mostrar as vantagens e desvantagens da arquitetura definida neste trabalho.
- Publicação de artigos em eventos científicos para avaliação da proposta durante todo o percurso do trabalho. Este processo foi de essencial importância para a evolução do trabalho, com sugestões, questões a serem melhor definidas e críticas para redefinições do trabalho quando necessário.

1.4 ORGANIZAÇÃO DA DISSERTAÇÃO

Este trabalho encontra-se estruturado conforme descrito a seguir. O Capítulo 2 apresenta a fundamentação teórica deste trabalho incluindo conceitos sobre CSCW, sistemas colaborativos móveis, desenvolvimento baseado em componentes e arquiteturas de referência. No Capítulo 2 também são apresentados diferentes trabalhos relacionados ao suporte ao desenvolvimento de sistemas colaborativos móveis. No Capítulo 3 são apresentados os requisitos genéricos de

colaboração móvel e de orientação a componentes, em que todos os requisitos foram encontrados por meio de uma extensa busca na literatura de trabalhos que visam definir requisitos para sistemas colaborativos que incluem aspectos da mobilidade. No Capítulo 4 é apresentada a arquitetura de referência. No Capítulo 5 são mostradas as implementações e testes (componentes e protótipos), além de uma comparação com o desenvolvimento utilizando a infraestrutura baseada no CReAMA e o desenvolvimento sem auxílio da arquitetura. Por fim, o Capítulo 6 conclui a dissertação e apresenta sugestões de possíveis trabalhos futuros.

2 CONTEXTUALIZAÇÃO E TRABALHOS RELACIONADOS

2 CONTEXTUALIZAÇÃO E TRABALHOS RELACIONADOS

Este Capítulo discorre sobre sistemas colaborativos e sistemas colaborativos móveis, mostrando definições utilizadas na literatura e caracterizando esses tipos de sistemas. Também são apresentados os tipos de suporte utilizados para desenvolver aplicações colaborativas móveis, que no presente trabalho foram agrupados em duas categorias: tecnológica (plataformas, toolkits, etc.) e metodológica (métodos, modelos de referência, organização ou agrupamento de requisitos, etc.). Além disso, é apresentada uma revisão da literatura no que diz respeito a componentes de software e componentização utilizada em sistemas colaborativos. É apresentado também a definição de arquitetura de referência utilizada no presente trabalho. Por fim, é feita uma discussão sobre os diferentes tipos de plataformas móveis e suas diferenças, apontando vantagens e desvantagens das principais tecnologias para desenvolvimento de software para dispositivos móveis.

2.1 SISTEMAS COLABORATIVOS MÓVEIS

A Web 2.0 aumentou a possibilidade de expressão por meio das ferramentas de comunicação mediada por computador e, consequentemente, algumas formas de interação entre usuários são ampliadas. De acordo com Merlo (2010), um desses serviços é a colaboração, que utiliza maneiras inovadoras de interação entre os usuários, proporcionando melhores resultados do trabalho colaborativo, tanto para fins pessoais ou profissionais.

O termo Sistemas Colaborativos é aceito como a tradução para os termos *Groupware systems* e CSCW (*Computer Supported Cooperative Work*) *systems* (MICHALSKY, 2012). Os sistemas colaborativos estão relacionados a sistemas computacionais para apoio à colaboração e têm sido estudados antes mesmo da Web (NICOLACI-DA-COSTA et al., 2011). De acordo com Ellis et al. (1991), um sistema colaborativo é um sistema apoiado por computador que oferece suporte a grupos de usuários engajados numa mesma tarefa e que provê uma interface para um ambiente compartilhado.

Muitos trabalhos na área de CSCW fazem referência ao modelo 3C, originalmente proposto por Ellis et al. (1991). No modelo 3C, a colaboração é analisada nas dimensões de comunicação, coordenação e cooperação. A comunicação envolve a troca de mensagens e a negociação de compromissos. Através da coordenação, as pessoas, as atividades e os recursos são gerenciados para lidar com conflitos e evitar a perda dos esforços de comunicação e de cooperação. A cooperação é a produção conjunta dos membros do grupo em um espaço compartilhado, gerando e manipulando objetos de cooperação na realização das tarefas (GEROSA, 2006).

É importante ressaltar que a área de CSCW é altamente interdisciplinar e os processos de negócio que definem a dinâmica do grupo de trabalho são difíceis de modelar e apoiar (GEROSA e FUKS, 2008). Ainda de acordo com Gerosa e Fuks (2008), o desenvolvimento de sistemas colaborativos requer programadores qualificados e treinados para lidar com protocolos, conexões, compartilhamento de recursos, distribuição, processamento e gerenciamento de sessão. Torna-se, portanto, atrativo o desenvolvimento de um conjunto de ferramentas que encapsule as complexidades de baixo nível, propiciando a investigação de interação para a construção desses sistemas por meio da prototipação.

Ainda no contexto de CSCW, existe um segmento específico denominado Colaboração Móvel Apoiada por Computador, que surgiu das novas oportunidades trazidas pelas tecnologias de comunicação sem fio e dispositivos de computação móvel (HERSKOVIC et al., 2011). A atual infraestrutura de comunicação e computação para dispositivos móveis tem incentivado o trabalho colaborativo (e a colaboração móvel de uma forma geral), ou seja, usuários trabalhando em conjunto nas tarefas e perseguindo um objetivo comum, mesmo que em lugares distintos (PICHILIANI e HIRATA, 2012). Para Neyem et al. (2007), o resultado é um novo cenário de colaboração em que a maioria das soluções tradicionais para problemas típicos de projetos de sistemas colaborativos não são aplicáveis.

Tecnologias móveis são vistas como facilitadoras de negócios e têm o potencial para apoiar a prática do trabalho colaborativo. Ao implementar uma tecnologia de colaboração móvel apoiada por computador, as organizações podem aumentar fortemente as possibilidades de interação. Por exemplo, permitir que seu pessoal de manutenção esteja sempre acessível para fornecer e receber

informações. Com isso, a equipe estará mais disponível para a manutenção planejada ou não planejada e agir o mais rápido possível (TSIRULNIK, 2009).

O ambiente de colaboração móvel apresenta uma ampla gama de uso da tecnologia, as práticas sociais e comportamentos inovadores e tem potencial para a exploração dos diferentes papéis que a cooperação, comunicação e coordenação fornecem. Além disso, os fabricantes de dispositivos e operadoras de telecomunicações têm proporcionado muitos recursos e oportunidades de negócios que motivam os desenvolvedores a criar, publicar e distribuir aplicativos móveis (PICHILIANI e HIRATA, 2012). Para Berkenbrock (2009), a imensa utilização de dispositivos móveis foi o que facilitou esse novo tipo de colaboração, dando origem a à nova geração de sistemas colaborativos móveis.

De acordo com Pichiliani e Hirata (2012), o suporte para a colaboração em tempo real que permite a interação entre os participantes distribuídos geograficamente é ainda um considerável desafio de engenharia de software, devido ao esforço necessário para apoiar adequadamente os requisitos de colaboração. O desenvolvimento de aplicações móveis para apoiar o trabalho colaborativo pode ser uma tarefa complexa, uma vez que trata-se de uma área recente e existem desafios que devem ser superados no que diz respeito ao processo de desenvolvimento, apoio à colaboração e aprovação dos usuários (HERSKOVIC et al., 2011).

A mobilidade dos usuários, a utilização de tecnologias de comunicação sem fio e o uso de dispositivos portáteis introduzem desafios que devem ser levados em consideração na concepção da infraestrutura de *software* dessas redes (BATISTA E SILVA, 2012). Ambientes móveis possuem uma série de restrições devido à mobilidade. Tecnologias de comunicação sem fio podem apresentar períodos de intermitência do sinal, baixa largura de banda e alta latência, limitando a comunicação nesses ambientes. Ainda de acordo com Batista e Silva (2012), além dos problemas citados, a perda de conexões por períodos curtos ou prolongados é frequente, tornando útil o suporte à realização de operações de forma desconectada pela infraestrutura de software.

Segundo Pichiliani e Hirata (2012), requisitos de colaboração incluem baixa latência de comunicação, percepção sobre o dispositivo, mecanismos de coordenação, técnicas de controle de concorrência, entre outros. Os problemas são agravados pelo fato da baixa disponibilidade de recursos em dispositivos móveis,

pois a utilização dos recursos de comunicação aumenta significativamente o consumo de energia nestes dispositivos que têm sua bateria limitada. Tais restrições apresentam grande impacto sobre a concepção e estrutura de aplicações nativas nesses ambientes e devem ser consideradas na elaboração da infraestrutura de comunicação (BATISTA E SILVA, 2012).

2.1.1 Suporte ao Desenvolvimento de Sistemas Colaborativos Móveis

O desenvolvimento de sistemas colaborativos móveis é uma atividade desafiadora, uma vez que frequentemente envolve a compreensão tanto de fatores técnicos quanto sociais. As estratégias de pesquisa para apoiar recursos de colaboração em aplicações móveis incluem kits de ferramentas e desenvolvimento baseado em componentes. Segundo Pichiliani e Hirata (2012), essas abordagens têm como objetivo reduzir o esforço de desenvolvimento de sistemas colaborativos móveis.

Uma questão importante é que a falta de uma abordagem sistemática, juntamente com a falta de requisitos específicos para a implementação de sistemas que apoiem a colaboração móvel, gera a necessidade de um suporte abrangente para orientar e organizar o processo de desenvolvimento de novas tecnologias móveis que atendem a todos os requisitos de colaboração (SYAFAR e GAO, 2013).

O suporte ao desenvolvimento de sistemas ou aplicações colaborativas móveis, geralmente é encontrado sob duas formas (MELOTTI et al., 2013): tecnológica (plataformas, toolkits, etc.) e metodológica (métodos, modelos de referência, organização ou agrupamento de requisitos, etc.). Nesta subseção, os trabalhos relacionados encontrados na literatura são apresentados a seguir seguindo essa distinção.

Dentre as soluções tecnológicas, algumas plataformas foram propostas para o desenvolvimento de sistemas colaborativos móveis. A plataforma MUSE, apresentada por Byrne (2011), fornece um sistema para a criação de aplicações de aprendizagem colaborativa. A plataforma contém três componentes principais: o servidor de aplicação MUSE, um conjunto de serviços de infraestrutura e o *middleware* MUSE. Como definido pelos autores, MUSE é uma plataforma MCSCL (*Mobile Computer Supported Collaborative Learning* – Aprendizagem Colaborativa

Apoiada por Computador ou Dispositivo Móvel) para aplicações multimídia que aborda as considerações técnicas de aplicações colaborativas móveis e pedagógicas da aprendizagem colaborativa. MUSE utiliza um modelo de arquitetura orientada a serviços para facilitar o reúso de componentes fracamente acoplados, possibilitando assim o desenvolvimento de aplicações flexíveis e reconfiguráveis, adequados a esse ambiente heterogêneo.

MUSE se concentra na produção de narrativa digital e fornece um conjunto de serviços genéricos para serem utilizados por desenvolvedores. Uma das contribuições desse trabalho é um conjunto de requisitos funcionais para plataformas MCSCL. Assim, desenvolvedores de aplicações para aprendizagem colaborativa podem se apoiar nesses requisitos para criar novas aplicações ou plataformas melhoradas. Um ponto importante é que, apesar do autor realizar uma ampla busca na literatura relacionada à colaboração móvel e ter encontrado requisitos genéricos para aplicações colaborativas móveis, o foco do trabalho está na aprendizagem colaborativa por meio de narrativa digital. A plataforma não busca cumprir todos os requisitos genéricos encontrados.

No trabalho de Bendel e Schuster (2012) é proposto um toolkit denominado *WatchMyPhone* (WMP) para facilitar a construção de aplicações colaborativas. Esse toolkit de desenvolvedor é focado na criação de aplicações de colaboração em dispositivos móveis, especialmente em aplicações de compartilhamento de interface gráfica de usuário (GUI), como visualizadores de texto em um ambiente multiusuário. É fornecido um conjunto de ferramentas, bem como uma aplicação de demonstração para edição de texto compartilhado baseada em Android.

O conjunto de ferramentas proposto pelo WMP é centrado no cliente e possibilita que o desenvolvedor do aplicativo se concentre na utilização de funcionalidades já desenvolvidas por meio dos chamados “componentes de desenvolvedor” (que são parte da GUI do sistema) e “componentes de suporte”, que fornecem funcionalidades como sincronização de dados e uso de protocolos de rede necessários para a colaboração (BENDEL E SCHUSTER, 2012). Apesar das diversas funcionalidades oferecidas nesse trabalho, ainda existem muitas questões em aberto, como o tratamento de problemas relacionados à conectividade e desconectividade do usuário (consistência e sincronização) e a falta de suporte a informações de rede, bateria e sensores.

Com o intuito de facilitar o desenvolvimento de groupware, Fonseca e Carrapatoso (2006) propõem uma arquitetura, denominada SAGA, que utiliza um modelo baseado em serviços. Nessa arquitetura, serviços genéricos são oferecidos para serem reutilizados em groupware, como por exemplo: controle de concorrência, notificação de eventos e autenticação. Entretanto, a arquitetura proposta não considera algumas questões de mobilidade, como a localização e como os usuários estão distribuídos geograficamente, nem tampouco considera a necessidade frequente das aplicações de groupware interoperarem entre plataformas distintas e tecnologias heterogêneas.

No que diz respeito ao suporte metodológico, destaca-se o trabalho de Herskovic *et al.* (2011), em que é apresentado um framework que especifica uma lista de requisitos gerais a serem considerados durante a concepção e projeto de um sistema de colaboração móvel a fim de aumentar a sua probabilidade de sucesso. O framework foi obtido com base em um levantamento de pesquisas relacionadas, bem como das experiências dos autores em desenvolvimento de sistemas colaborativos móveis.

A literatura oferece muitas perspectivas sobre os aspectos importantes a serem considerados no desenvolvimento de sistemas colaborativos móveis. Mas de acordo com Herskovic *et al.* (2011), a maioria desses estudos está focada em uma área específica de aplicação e não considera o impacto dos requisitos gerais sobre o sucesso do projeto de desenvolvimento, nem como relacionar os requisitos com a maneira como o trabalho móvel ocorre. A ideia principal no trabalho apresentado é fornecer uma visão mais abrangente das necessidades típicas de colaboração móvel e as relações entre elas.

Ainda em relação ao suporte metodológico, o trabalho apresentado por Berkenbrock et al. (2009) explora requisitos específicos no que diz respeito a interfaces gráficas no projeto de sistemas colaborativos móveis. Esse trabalho também apresenta técnicas e abordagens para a construção e verificação de requisitos para interfaces gráficas.

Os trabalhos de Neyem et al. (2008) e Herskovc et al. (2008) apresentam diferentes tipos de suporte metodológico para workspaces móveis e compartilhados. O primeiro propõe um processo de desenvolvimento e várias instruções baseadas em revisão de literatura e experiência dos autores para o desenvolvimento desses

sistemas. O segundo apresenta um framework com requisitos gerais que normalmente estão presentes nesses sistemas.

Trabalhos como (NEYEM et al., 2008), (HERSKOVIC et al., 2008) e (BERKENBROCK et al., 2009) também exploram alguns requisitos genéricos para colaboração móvel, assim como apresentam requisitos de domínios e cenários específicos. Esses trabalhos apoiaram a definição de CReAMA, porém, o framework apresentado por Herskovic *et al.* (2011) encontra-se como principal base para a organização dos requisitos, que serão descritos no próximo Capítulo, devido à visão mais abrangente para sistemas colaborativos móveis que esse trabalho apresenta.

2.2 DESENVOLVIMENTO BASEADO EM COMPONENTES DE SOFTWARE

A ideia de componentes de software e reúso não é recente. Durante a *NATO Software Engineering Conference*, em 1968, McIlroy (1968) em seu artigo "*Mass Produced Software Components*" introduziu o conceito de componentes de software e reúso (MICHALSKY, 2012). Componente de software é o termo utilizado para descrever uma unidade independente de software que encapsula uma série de funcionalidades, pode ser combinado com outros componentes e deve ser reusável em diferentes sistemas.

Na busca de novas metodologias e práticas de desenvolvimento de software, algumas soluções como a orientação a objetos e padrões de projeto, ainda deixam distante o sonhado reúso que poderia ser utilizado para uma implementação mais rápida de sistemas computacionais. Os sistemas estão a cada dia mais complexos e, muitas vezes, os padrões de desenvolvimento e gerenciamento de processos não atendem as expectativas desejadas. A componentização apresenta-se como uma alternativa capaz de lidar com problemas ligados ao reaproveitamento, não somente de código, mas também de funcionalidades e interfaces e tempo de desenvolvimento.

Segundo Szyperski et al. (2002), componentes de software são desenvolvidos objetivando o reúso em diversos sistemas, reduzindo o investimento inicial e os custos de manutenção. Também há uma preocupação com reúso de

novos componentes gerados no processo de desenvolvimento da aplicação (MICHALSKY, 2012).

Para Wulf et al. (2008), sistemas baseados em componentes oferecem duas formas de refazer ou re-projetar um software: por meio da substituição de componentes e por meio de programação (alteração de um componente existente e criação de um novo componente). Com base no uso de componentes, o sistema pode ser estendido para acompanhar a evolução dos processos de trabalho, das necessidades e da experiência com o uso do sistema (GEROSA, 2006).

É importante mencionar que a definição de componentes é ampla e variável, no contexto do desenvolvimento de software, um componente pode referir-se a uma aplicação inteira ou mesmo a uma classe (MICHALSKY, 2012). Oliveira e Gerosa (2011) fizeram um levantamento sobre as definições de componente de software e o definem como: um módulo autocontido com interfaces explícitas e bem definidas, que pode ser reusado, substituído e composto com outros componentes.

Para Gerosa e Steinmacher (2011), o conceito de componente é similar ao conceito de módulo no sentido que ambos dividem os produtos em partes menores, coesas e fracamente acopladas, mas diferem dos módulos pelo fato de que os componentes precisam ser “encaixados”. Além disso, diferentemente dos módulos, os componentes seguem uma série de padrões, possuem uma infraestrutura específica para o gerenciamento e a execução e são encapsulados em uma unidade executável.

Em abordagens típicas de desenvolvimento de Software, componentes só são visíveis em tempo de projeto e a composição é feita por meio de uma linguagem de programação ou por um construtor visual em tempo de projeto (WULF et al., 2008). Uma vez que essas abordagens não têm o objetivo de permitir que qualquer alteração da estrutura do componente seja feita em tempo de execução, a possibilidade do re-agrupamento é perdido após a compilação.

Para projetar um sistema baseado em componentes é necessário entender os benefícios e dificuldades da tecnologia, além do ferramental disponível. Os benefícios da componentização estão ligados a manutenibilidade, reúso, composição, extensibilidade, integração e escalabilidade (D’SOUZA e WILLS, 1998). Ainda em relação ao projeto, é necessário que os componentes sigam um padrão de acoplamento definido por um modelo de componentes. Um modelo de componentes

também define padrões de interoperabilidade para a comunicação entre componentes escritos em diferentes linguagens ou implantados em diferentes máquinas, além de definir também padrões de composição, padrões de evolução de componentes e padrões de instalação (GEROSA e STEINMACHER, 2011). Para Gerosa (2006), classificar algo como componente depende também de como o código foi concebido, construído e como ele se relaciona com o restante do sistema e não somente da aderência a uma padronização.

Vale ressaltar que a documentação dos componentes é muito importante. Além da documentação gráfica da estrutura e dos inter-relacionamentos entre os componentes, é necessário descrevê-los textualmente. Nessa descrição, é necessário englobar aspectos referentes aos requisitos funcionais e não-funcionais (GEROSA, 2006). Paludo e Burnett (2005) propõem a utilização da estrutura de documentação de padrões de projeto para documentar componentes. É documentado, por exemplo, nome, intenção, aplicabilidade, variabilidade, estrutura, código de exemplo, usos conhecidos e componentes relacionados.

O uso de componentes apresenta diversas vantavens e desvantagens. Segundo Gerosa (2006), com componentes são reduzidas às interdependências fixas no código fonte da aplicação, melhorando a extensibilidade e facilitando a adaptação e extensão. De uma forma geral, podemos incluir como vantagens: melhor capacidade de adaptação, melhor suporte à prototipação e evolução de software, substituição dinâmica de partes do sistema, gerenciamento de mudanças, interoperabilidade, entre outras (GEROSA e STEINMACHER, 2011).

As desvantagens incluem a dificuldade do desenvolvimento de novos componentes e dificuldades do desenvolvimento com componentes já existentes. De acordo com Michalsky (2012), as dificuldades são separadas em dificuldades do desenvolvimento para componentização e dificuldades do desenvolvimento com componentização. As dificuldades do desenvolvimento para componentização estão ligadas ao esforço inicial de análise, projeto e desenvolvimento da infraestrutura para o reúso, e as dificuldades do desenvolvimento com componentização estão ligadas ao esforço de entendimento dos componentes e das ferramentas envolvidas, à perda de flexibilidade, à dependência de terceiros e à adaptação do processo de desenvolvimento.

Muitas vezes, é difícil encontrar um componente que atenda plenamente às funcionalidades desejadas, além da chance de incorporar erros de software produzidos por terceiros. Então, algumas vezes é necessário muito esforço para produzir uma grande quantidade de código com a ideia de alterar esses componentes (GEROSA e STEINMACHER, 2011).

As abordagens de desenvolvimento de software baseado em componentes têm sido o foco de muitos esforços de pesquisa, desde a substituição de componentes, até a criação de novos frameworks, plug-ins e arquiteturas. De acordo com Pichiliani e Hirata (2012), essas abordagens representam o estado da arte de técnicas de Engenharia de Software que aliviam ou diminuem o esforço exigido para o desenvolvimento de aplicações móveis e colaborativas.

2.2.1 Desenvolvimento de Groupware baseado em Componentes

No que diz respeito ao desenvolvimento baseado em componentes para groupware, o GroupKit (ROSEMAN e GREENBERG, 1992) oferece widgets para compor a interface gráfica, facilitando o tratamento de conexões, gerenciamento de eventos e o compartilhamento de dados. A ideia principal é reduzir efetivamente a complexidade de sistemas groupware por meio de de elementos chave oferecidos pelo GroupKit.

O SDGToolKit (TSE e GREENBERG, 2004) é derivado do GroupKit e fornece funcionalidades de alto nível para apoiar o trabalho de grupos presenciais usando um monitor compartilhado. Uma das principais características do SDGToolkit é a gerência automática de vários mouses e teclados em uma mesma tela para facilitar o trabalho colaborativo presencial em sistemas groupware.

Ainda em relação ao desenvolvimento baseado em componentes para groupware, o Groupware Workbench (GW) é um projeto de software livre, desenvolvido principalmente em parceria entre o IME/USP, a UFES, a UESB e a PUC-Rio e é hospedado no Centro de Competência em Software Livre (CCSL) do IME/USP (GROUPWARE WORKBENCH, 2013). O projeto objetiva oferecer um ferramental baseado em componentes na Web 2.0, de modo a encapsular as complexidades técnicas envolvidas para o desenvolvimento de sistemas

colaborativos para que o desenvolvedor não precise conhecer os detalhes de implementação dos componentes.

Na abordagem da plataforma GW, os componentes são baseados no modelo 3C de colaboração, no qual a colaboração é analisada a partir das dimensões de comunicação, coordenação e cooperação (GEROSA e FUKS, 2008). A tecnologia utilizada no GW oferece suporte ao padrão MVC, à propagação de eventos, à persistência de dados e à criação de elementos de interface. Os componentes chamados de *collablets* são utilizados de forma hierárquica para construir as ferramentas de colaboração (MAMANI e GEROSA, 2011).

O modelo de componentes do GW provê suporte à instalação, atualização, agrupamento, customização, disponibilização, reúso, interdependências e ciclo de vida dos componentes (MELOTTI e GOMES, 2012). Segundo Oliveira (2010), o GW foi desenvolvido para favorecer aspectos da componentização a fim de prototipar o apoio à colaboração. Projetos desenvolvidos no ambiente do GW, como o Arquigrafia Brasil[3], utilizam algumas tecnologias como o padrão REST e dados nos formatos XML e JSON.

Uma limitação atual do GW é o fato deste não oferecer suporte a clientes móveis para os sistemas colaborativos desenvolvidos com base nele (MELOTTI e GOMES, 2012). Devido à grande popularização da computação móvel, que introduziu novos cenários de colaboração e novas maneiras de trabalhar em equipe, surge a necessidade de trazer esse ferramental para o desenvolvimento de sistemas colaborativos móveis.

2.3 ARQUITETURA DE REFERÊNCIA

A concepção de uma arquitetura de referência é entendida neste trabalho como uma forma de apresentar um padrão genérico para um projeto (ZAMBIASI, 2012). Para compor uma arquitetura de referência é necessário apresentar os tipos

[3] http://www.arquigrafia.org.br/

dos elementos envolvidos, como eles interagem e o mapeamento das funcionalidades para estes elementos (HOFMEISTER et al., 2000). De maneira geral, uma arquitetura de referência deve abordar os requisitos para o desenvolvimento de soluções, guiado pelo modelo de referência e por um estilo arquitetural de forma a atender as necessidades do projeto (MACKENZIE et al., 2009).

Inicialmente, com o modelo de referência e suas funcionalidades, uma arquitetura de referência tem por objetivo fornecer um norteador para o mapeamento das funcionalidades do modelo de referência para um modelo que deve se formar um sistema, ou seja, a arquitetura é o mapeamento do modelo de referência para elementos de software (BASS et al., 2003). Segundo MacKenzie et al. (2006), ela é, em si, um padrão genérico para um projeto e deve abordar requisitos para o desenvolvimento de soluções, guiadas pelo modelo de referência, de forma a atender as necessidades do projeto.

Junto com o modelo de referência, a arquitetura também é guiada por um estilo arquitetônico (ou arquitetural), formando um conjunto de requisitos e características de norteamento (ZAMBIASI, 2012). Dessa maneira, se torna necessário apresentar os tipos dos elementos envolvidos, como eles interagem e o mapeamento das funcionalidades desses elementos, com a finalidade de compor a arquitetura de referência (HOFMEISTER et al., 2000). Vale ressaltar que não foi encontrada na literatura, uma proposta de arquitetura de referência para guiar o desenvolvimento de sistemas colaborativos móveis.

De acordo com Bass et al. (2003), a arquitetura de software de um sistema computacional refere-se à sua estrutura, consistindo de elementos de software, propriedades externamente visíveis desses elementos e os relacionamentos entre eles. A arquitetura define elementos de software (ou módulos) e envolve informações sobre como eles se relacionam uns com os outros. Uma arquitetura pode envolver mais de um tipo de estrutura, com diferentes tipos de elementos e de relacionamentos entre eles. A arquitetura omite certas informações sobre os elementos que não pertencem às suas interações.

Diferente do que foi comentado no parágrafo anterior, uma Arquitetutra de Referência não define um sistema computacional específico, e nesse aspecto uma Arquitetura de Software “comum”, como definido por Bass et al. (2003) se diferencia

de uma Arquitetura de Referência, que específica requisitos e elementos de software que serão utilizados na criação de outros sistemas.

Conforme discutido anteriormente, a reutilização é um aspecto-chave para se obter um bom projeto de software. No que refere ao reúso no projeto da arquitetura de software, estilos e padrões arquitetônicos são importantes ferramentas. Muitos autores consideram estilos e padrões como diferentes termos para designar o mesmo conceito, tal como Gorton (2006). Outros consideram estilos e padrões conceitos diferentes, como é o caso de Albin (2003). Neste texto, vamos tratá-los como conceitos distintos.

Um estilo arquitetônico define um vocabulário de tipos de elementos e tipos de relacionamentos, e um conjunto de restrições sobre como eles podem ser combinados (SHAW & GARLAN, 1996). Padrões arquitetônicos também estabelecem tipos de elementos e de relacionamentos entre eles, mas sua apresentação segue uma forma bem definida, indicando nome, contexto, problema, solução e consequências. Especialmente a solução define estratégias para tratar o problema, o que não acontece com os estilos arquitetônicos. Assim, normalmente, estilos têm uma apresentação mais livre e são descritos de maneira mais abstrata que padrões arquitetônicos.

Vale mencionar que neste trabalho, a arquitetura de referência segue um estilo arquitetônico em camadas. No estilo em camadas, um sistema é organizado hierarquicamente, como um conjunto ordenado de camadas, cada uma delas construída em função das camadas abaixo e fornecendo serviços para as camadas acima. O conhecimento é unidirecional, por exemplo, uma camada conhece as camadas abaixo, mas não tem qualquer conhecimento sobre as camadas acima. Há uma relação cliente-servidor entre uma camada usuária de serviços e suas camadas inferiores, provedoras de serviços (BLAHA & RUMBAUGH, 2006). Cada camada acrescenta um nível de abstração sobre a camada inferior (MENDES, 2002).

2.4 PLATAFORMAS MÓVEIS

O desenvolvimento de aplicações para dispositivos móveis está cada vez mais difundido na indústria e na academia, e entre as plataformas utilizadas podemos citar o Black Berry OS, Windows Phone, Android, iOS, etc. Dessas plataformas existentes para aplicações móveis, Android (Google) e iOS (Apple) juntas detiam cerca de 91.1% do mercado em 2013[4]. Entre os critérios mais comuns na escolha da plataforma estão: público alvo da aplicação; afinidade dos desenvolvedores com a plataforma; ferramentas e linguagens e custo de licenças de hardware para o desenvolvimento. Segundo Mascarenhas et al. (2013), esses e outros fatores quando desconsiderados podem prejudicar o desenvolvimento, a evolução e a manutencão da aplicação.

No trabalho desenvolvido por Goadrich and Rogers (2011), os autores apresentam uma análise comparativa de desenvolvimento entre as plataformas móveis Android e iOS. A principal analisada foi qual plataforma o setor acadêmico deve ensinar para alunos de graduação. Os autores comparam as plataformas com exemplos práticos e questões sobre hardware, interface gráfica e SDKs. Os autores concluem o trabalho apontando que o desenvolvimento iOS em relação ao Android pode ser mais difícil devido ao hardware e ferramentas específicas e a linguagem de programação menos conhecida.

No trabalho apresentado por Ribeiro et al. (2011), é comparado o desenvolvimento de aplicações iOS e Android apresentando detalhes de cada plataforma, como ambiente de desenvolvimento, linguagem de programação e documentação. Os autores concluem que os pontos fortes do Android são: maior difusão da linguagem de programação Java em relação ao Objective-C do iOS; o emulador dispõe de mais funcionalidades; e a licença ser aberta possibilita a criação de aplicações mais robustas. Já os pontos fortes do iOS em relação ao Android são:

[4] http://www.idc.com/getdoc.jsp?containerId=prUS24442013

o seu ambiente de desenvolvimento (XCode) é melhor integrado e possui ferramentas de interface superiores.

No que diz respeito ao ambiente de desenvolvimento, o Android provê maior flexibilidade quanto ao ambiente de desenvolvimento. Enquanto o desenvolvimento para Android é possível utilizando-se os sistemas operacionais Windows, Linux e MacOS, o desenvolvimento para iOS só é possível utilizando-se o MacOS. Isto leva também à necessidade de um hardware específico já que apenas os computadores desenvolvidos pela Apple suportam o sistema operacional MacOS. Outra questão é a IDE, para desenvolvimento iOS só é possível utilizar a ferramenta XCode, já no desenvolvimento Android, o desenvolvimento pode ser feito com várias IDEs (MASCARENHAS et al., 2013).

Depois que a aplicação estiver pronta, é preciso distribuir para os usuários. Existem particularidades de cada plataforma que são muito importantes e devem ser levadas em consideração. A Tabela 1, apresentada no trabalho de Mascarenhas et al. (2013), indica que o Android possui vantagens no que diz respeito à distribuição de aplicações. Essas vantagens incluem a liberdade para distribuir o produto da forma que quiser e o preço para distribuir no canal oficial de distribuição.

TABELA 1 – COMPARAÇÃO DE PLATAFORMA MÓVEIS

Critério	**Android**	**iOS**
Fabricantes de dispositivos	Mais de 50	1
Tipos de dispositivos	Mais de 1500	10
Canal de distribuição oficial	Google Play	Apple Store
Distribuição standalone	Sim	Não
Licença de desenvolvedor	$25 (taxa única)	$99 (por ano)

2.5 CONSIDERAÇÕES SOBRE O CAPÍTULO

Neste Capítulo foram apresentados os tipos de suporte que podem ser utilizados para desenvolver aplicações colaborativas móveis, sendo que no presente trabalho eles foram agrupados em duas categorias: tecnológica (plataformas,

toolkits, etc.) e metodológica (métodos, modelos de referência, organização ou agrupamento de requisitos, etc.). O desenvolvimento baseado em componentes apresenta a vantagem de melhor prototipação de software depois que os componentes estiverem prontos. Então uma possibilidade é utilizar ToolKits de componentes junto com alguma abordagem metodológica como métodos ou modelos para o desenvolvimento desses sistemas.

Além disso, foi apresentada também uma revisão da literatura no que diz respeito à arquitetura de referência e componentes de software, ressaltando-se apresentado as vantagens desse tipo de ferramental. Então utilizando os benefícios da componentização que incluem melhor capacidade de adaptação, melhor suporte à prototipação e evolução do software, substituição dinâmica de partes do sistema juntamente com uma arquitetura de referência, provendo um modelo padrão para o desenvolvimento desses componentes, será atingida a meta de uma boa alternativa para o desenvolvimento de sistemas não triviais, como sistemas colaborativos.

Devido à grande popularização da computação móvel, que introduziu novos cenários de colaboração e novas maneiras de trabalhar em equipe, surge a necessidade de trazer esse ferramental para o desenvolvimento de sistemas colaborativos móveis na plataforma Android, que é a plataforma que apresentou mais vantagens, além de estar entre as duas mais populares atualmente no mercado.

3 REQUISITOS DA ARQUITETURA DE REFERÊNCIA

3 REQUISITOS DA ARQUITETURA DE REFERÊNCIA

A arquitetura de referência proposta tem como principal objetivo orientar o desenvolvimento de sistemas colaborativos móveis baseados em componentes. Sistemas desenvolvidos de acordo com essa arquitetura devem dar suporte ao desenvolvimento de componentes e à criação de aplicações colaborativas por meio da composição desses componentes. As aplicações e componentes são desenvolvidos para plataformas móveis, facilitando o uso de recursos inerentes a essas plataformas, tais como informações de sensores embarcados.

Com base na arquitetura de referência, o desenvolvedor poderá ser guiado para criar componentes e compor novas aplicações seguindo os padrões estabelecidos. Por exemplo, será possível construir toolkits que forneçam componentes para um domínio específico.

É importante ressaltar que a arquitetura foi definida considerando-se: aspectos da plataforma móvel, de sistemas colaborativos e da própria orientação a componentes. Com relação à plataforma móvel, neste trabalho optou-se por uma plataforma específica, visando-se a definição de uma arquitetura otimizada para as características da respectiva plataforma. A plataforma escolhida foi o Android, visto que se trata de uma plataforma de código aberto que vem sendo adotada por uma quantidade crescente de usuários e desenvolvedores em todo o mundo (VISION MOBILE, 2013).

A arquitetura proposta dará suporte ao desenvolvimento de novos sistemas baseados em componentes, considerando também aspectos relativos à comunicação com a Web (persistência, sincronização e disponibilidade de dados a partir de um serviço). Nas próximas subseções são descritos os requisitos considerados na definição da arquitetura e a origem desses requisitos. Para começar, definimos um cenário hipotético que representa uma gama de situações às quais os sistemas desenvolvidos a partir da arquitetura de referência devem dar suporte.

3.1 CENÁRIO MOTIVADOR

Para descrever o cenário motivador, chamaremos de Julian o desenvolvedor de componentes e de aplicações e consideraremos que Leonan será um usuário da aplicação colaborativa que será desenvolvida por Julian. É importante ressaltar que o cenário já prevê um ferramental (infraestrutura e componentes) construído com base na arquitetura proposta. Um cenário parecido foi desenvolvido e apresentado anteriormente em trabalho prévio (MELOTTI et al., 2013). Vale mencionar que o cenário foi construído com a motivação de que na literatura não existe um exemplo de uso e nenhum cenário fictício parecido para expor requisitos de sistemas orientados à componentes. Abaixo são numerados os conjuntos de passos descritos no cenário.

1. Julian precisa criar uma aplicação colaborativa para a empresa em que trabalha.
2. Essa aplicação será executada em um dispositivo móvel com o sistema operacional Android, para possibilitar acesso móvel a um sistema Web de sua empresa.
3. Julian faz a instalação da infraestrutura, junto com a qual são instalados os componentes utilizados para o desenvolvimento de aplicações colaborativas.
4. Ele inicia uma busca por componentes que possam ser utilizados para compor sua aplicação. É utilizado um catálogo disponibilizado junto com a infraestrutura.
5. São encontrados componentes que podem ser úteis a Julian e, seguindo o modelo de componentes, ele começa a compor a aplicação, customizando os componentes de acordo com as necessidades.
6. Julian percebe que precisará criar novos componentes para sua aplicação. Seguindo o modelo de componentes e instruções fornecidas para a utilização da infraestrutura, ele cria os novos componentes.
7. Com os novos componentes criados, eles são disponibilizados em um repositório, para que outros desenvolvedores possam utilizá-los no futuro.

8. Julian compõe sua aplicação e faz seu upload por meio de um serviço web da empresa, junto com informações sobre a versão da aplicação e componentes utilizados.
9. Com o intuito de utilizar a aplicação, Leonan faz o download da aplicação e a instala em seu smartphone.
10. Os componentes são instalados junto com a aplicação, assim como as configurações de banco de dados e outros elementos necessários. Isso é feito de forma transparente, sem a necessidade de o usuário interferir.
11. Leonan agora pode utilizar a aplicação quando achar necessário. Sempre que Leonan está utilizando a aplicação, e consegue acesso à internet, a aplicação verifica se houve alguma atualização.
12. Eventualmente, Julian percebe que precisa fazer mudanças nos componentes ou na aplicação. Depois de fazer todas as atualizações, Julian faz o upload da aplicação modificada e registra a mudança da versão.
13. Quando a aplicação verifica a versão e não é a mesma do dispositivo, é bloqueado o acesso ao restante da aplicação. A aplicação fornece a opção de fazer o download e reinstalação da aplicação para o usuário.
14. Leonan faz o download e instala novamente a aplicação.
15. Leonan agora pode utilizar a aplicação quando achar necessário.

Esse cenário descreve como é feito o uso dos componentes pelo desenvolvedor de aplicações colaborativas, abordando desde a instalação da infraestrutura e dos componentes, até questões como a atualização de versões, mesmo depois que um usuário já estiver utilizando a aplicação.

3.2 DEFINIÇÃO DE REQUISITOS

A definição da arquitetura foi realizada em grande parte com base em um conjunto de requisitos definidos na literatura e apresentados em diversos trabalhos. Visto que se trata de uma arquitetura de referência para orientar o desenvolvimento de sistemas colaborativos móveis baseados em componentes, os requisitos foram organizados de acordo com os dois principais aspectos contemplados pela

arquitetura: requisitos de colaboração móvel e requisitos de desenvolvimento baseado em componentes.

3.2.1 Colaboração Móvel

Os requisitos de sistemas colaborativos móveis levantados com base na literatura podem ser agrupados nas seguintes categorias: autonomia (usuário e aplicação), proteção, comunicação, heterogeneidade, awareness, suporte, hardware e interface com usuário. Esses requisitos foram obtidos por meio de uma extensa busca por trabalhos na área, e são apresentados a seguir:

(A) AUTONOMIA DE USUÁRIO

i. Detecção automática de usuário (BERKENBROCK, DA SILVA e HIRATA, 2009), (BERKENBROCK et al., 2009), (HERSKOVIC et al., 2008), (HERSKOVIC et al., 2011), (NEYEM et al., 2008): o sistema deve coletar automaticamente e armazenar informações relacionadas à disponibilidade dos usuários.

Observações: Com base nessas informações de contexto, o sistema colaborativo possibilita o início de atividades colaborativas sob demanda.

ii. Colaboração sob demanda (HERSKOVIC et al., 2008), (HERSKOVIC et al., 2011), (NEYEM et al., 2008), (OCHOA et al., 2007), (PINELLE e GUTWIN, 2006): O sistema deve possibilitar aos participantes trabalharem desconectados na maioria do tempo e, se for o desejado, mudar para o uso online sob demanda.

(B) AUTONOMIA DE APLICAÇÃO

i. Variação do contexto (NEYEM et al., 2008): O sistema deve considerar automaticamente mudanças de contexto, a fim de proporcionar um apoio eficaz para a colaboração.

Observações: Alguns atributos relacionados ao contexto, como acesso a internet, acesso a servidores, indisponibilidade de serviço de GPS, etc., mudarão de um lugar para outro.

ii. Descoberta de serviço e dispositivo (HERSKOVIC et al., 2011): O sistema deve possibilitar que serviços disponíveis e dispositivos (como telas públicas, *Smartboards*, upload de arquivos, etc.) sejam dinamicamente detectados e integrados no ambiente de colaboração em torno do contexto de trabalho.

Observações: exemplos de serviços e dispositivo são telas públicas, Smartboards, serviço de upload de arquivo, entre outros.

iii. Autonomia (BERKENBROCK, DA SILVA e HIRATA, 2009), (BERKENBROCK et al., 2009), (NEYEM et al., 2008), (PINELLE e GUTWIN, 2006): O sistema deve funcionar autonomamente para considerar mudanças no contexto do aparelho e ambiente.

Observações: por exemplo, a aplicação pode perceber quando a bateria estiver acabando, com o intuito de economizá-la para tarefas essenciais.

(C) PROTEÇÃO

i. Sessões de trabalho (HERSKOVIC et al., 2008), (HERSKOVIC et al., 2011), (NEYEM et al., 2008): o sistema deve garantir que a interação entre os usuários móveis seja protegida de modo a evitar a participação não autorizada no grupo e invalidar o acesso a recursos compartilhados entre eles.

ii. Privacidade do usuário (HERSKOVIC et al., 2008), (HERSKOVIC et al., 2011): o sistema deve permitir que cada usuário seja capaz de escolher quais dados compartilhar e, algumas ações a serem realizadas em particular.

iii. Segurança (HERSKOVIC et al., 2008), (HERSKOVIC et al., 2011), (NEYEM et al., 2008): O sistema deve possibilitar que o trabalho de cada usuário seja protegido para que ninguém possa destruir o trabalho de outro.

(D) COMUNICAÇÃO

i. Mensagens síncronas (HERSKOVIC et al., 2008), (HERSKOVIC et al., 2011): O sistema deve permitir a troca de mensagens quando dois usuários estiverem disponíveis simultaneamente e acessíveis.

ii. Mensagens assíncronas (HERSKOVIC et al., 2008), (HERSKOVIC et al., 2011): O sistema deve possibilitar-lhes enviar mensagens que eles recebem quando ficarem online, no caso de dois usuários que trabalham em diferentes momentos.

iii. Transferência de arquivos (HERSKOVIC et al., 2008), (HERSKOVIC et al., 2011): O sistema deve permitir a troca de arquivos entre usuários.

Observações: Membros de um determinado grupo de trabalho mantêm, localmente, os recursos que são relevantes para a realização de suas tarefas e em determinados momentos pode ser necessário compartilhar esses arquivos com outros usuários.

iv. Notificações (BERKENBROCK, DA SILVA e HIRATA, 2009), (BERKENBROCK et al., 2009): O sistema deve possibilitar notificações e mensagens para usuários.

Observações: As mensagens são entregues aos usuários móveis no momento em que se conectam ao servidor/serviço. Normalmente, uma janela pop-up é exibida na interface do usuário para mostrar as mensagens pendentes.

(E) HETEROGENEIDADE

i. Heterogeneidade (HERSKOVIC et al., 2008), (HERSKOVIC et al., 2011): O sistema deve ser utilizado em diferentes plataformas e dispositivos com diferentes recursos de hardware.

Observações: Colaboração pode envolver dispositivos heterogêneos, tais como laptops, smartphones e *tablets*. Esses dispositivos possuem recursos de hardware e capacidades diferentes de computação.

ii. Interoperabilidade (HERSKOVIC et al., 2008), (HERSKOVIC et al., 2011): O sistema deve entender significado de dados e serviços, mesmo que esses recursos sejam projetados por vários provedores distintos.

(F) AWARENESS

i. Percepção online (BERKENBROCK, DA SILVA e HIRATA, 2009), (BERKENBROCK et al., 2009), (HERSKOVIC et al., 2008), (HERSKOVIC et al., 2011), (NEYEM et al., 2008): O sistema deve ser capaz de mostrar informações sobre os elementos de colaboração do próprio sistema.

Observações: Exemplos de percepção online são listas de usuários conectados, locais de usuários, e atividades atuais desenvolvidas por usuários.

ii. Percepção offline (BERKENBROCK, DA SILVA e HIRATA, 2009), (BERKENBROCK et al., 2009), (HERSKOVIC et al., 2008), (HERSKOVIC et al., 2011): O sistema deve ser capaz de mostrar informações sobre os elementos de colaboração do próprio sistema que rem relação à interações já ocorridas no sistema, visíveis mesmo que os usuários encontrem-se off-line no momento da interação.

Observações: Alguns exemplos são a última modificação feita em um documento e autoria de texto.

iii. Percepção de transição (BERKENBROCK, DA SILVA e HIRATA, 2009), (BERKENBROCK et al., 2009), (HERSKOVIC et al., 2011): O sistema deve mostrar informações sobre as transições entre conexão e desconexão, ou a percepção sobre a entrega de mensagens.

(G) SUPORTE

i. Caching (BERKENBROCK, 2009), (HERSKOVIC et al., 2011): O sistema deve permitir que os dados compartilhados sejam automaticamente salvos na aplicação cliente, quando os usuários colaboram.

Observações: Esse requisito tem o objetivo de fornecer a cada usuário as informações mais atualizadas para trabalhar de forma autônoma (offline) quando necessário.

ii. Resolução de conflitos (BERKENBROCK, DA SILVA e HIRATA, 2009), (BERKENBROCK, 2009), (BERKENBROCK et al., 2009), (HERSKOVIC et al., 2008), (HERSKOVIC et al., 2011), (LUKOSCH, 2008), (NEYEM et al., 2008): O sistema deve permitir a correção das inconsistências nos dados compartilhados.

Observações: Os usuários móveis podem atualizar as informações locais sobre a aplicação móvel colaborativa quando estão trabalhando offline.

(H) HARDWARE

i. Uso de recursos (BERKENBROCK, DA SILVA e HIRATA, 2009), (BERKENBROCK et al., 2009), (NEYEM et al., 2008): O sistema deve levar em conta os recursos limitados de hardware, como capacidade de armazenamento e memória, poder de processamento, tamanho da tela de entrada de dados e vida útil da bateria.

ii. Limitações de rede (BERKENBROCK, DA SILVA e HIRATA, 2009), (BERKENBROCK et al., 2009): O sistema deve considerar as limitações de rede, como as conexões, que podem variar significativamente (em termos de atraso, largura de banda e taxa de erro).

(I) INTERFACE COM O USUÁRIO

i. Scrolling (BERKENBROCK, DA SILVA e HIRATA, 2009):

Observações: Devido ao espaço limitado na tela em dispositivos portáteis, muitas aplicações utilizam rolagem (scrolling) como um mecanismo para acessar informações que não cabem na tela. No entanto, a rolagem de tela para cima e para baixo, pode ser desconfortável para os usuários. Assim, é importante fazer o desenvolvimento de interfaces evitando-se a rolagem.

ii. Navegação (BERKENBROCK, DA SILVA e HIRATA, 2009): O sistema deve ser construído favorecendo a usabilidade e simplificando a navegação.

Observações: Pequenas telas podem tornar tarefas básicas, como navegação, uma tarefa difícil. Então, é importante definir uma estrutura de navegação adequada para apresentar melhor as informações, reduzindo o número de passos necessários para executar uma tarefa para simplificar a navegação.

iii. Capacidade de Gerenciamento (BERKENBROCK, DA SILVA e HIRATA, 2009), (BERKENBROCK et al., 2009): O sistema deve simplificar os passos para gerenciar as sessões e permitir que as tarefas sejam realizadas intuitivamente, pois um dos obstáculos em termos de usabilidade é a dificuldade de iniciar uma sessão.

3.2.2 Desenvolvimento Baseado em Componentes

No que diz respeito aos requisitos de desenvolvimento de software baseado em componentes, alguns deles são muito importantes para o decorrer do presente trabalho. Esses requisitos foram obtidos por meio de estudos de trabalhos sobre plataformas e frameworks orientados a componentes para sistemas colaborativos, como Gerosa (2006), Michalsky (2012) e Wulf et al. (2008), e para sistemas colaborativos móveis, como apresentado por Alaya et al (2011) e Martins et al. (2000). Também foi considerado o cenário motivador descrito anteriormente neste trabalho. Os requisitos levantados foram:

(J) COMPONENTIZAÇÃO

i. A instalação de componentes é vista em dois níveis diferentes: (a) instalação de componentes para que desenvolvedores possam compor aplicações, e (b) instalação de componentes dentro da aplicação para que usuários possam utilizá-los. O modelo de componentes deve dar suporte a ambos.

ii. A maneira como é feita a utilização dos componentes deve estar devidamente documentada, escrita de maneira clara, utilizando diagramas e

exemplos para facilitar o entendimento. Assim, outros desenvolvedores poderão usar os componentes para criar suas próprias aplicações.

iii. Um dos conceitos mais importantes em termos de componentização é o agrupamento de componentes. Esse agrupamento pode ser visto de duas maneiras: diferentes componentes gráficos agrupados em uma mesma tela, e o agrupamento lógico que diz respeito às relações e dependências entre os componentes. A maneira como esses dois tipos de agrupamento são feitos deve estar devidamente documentada no modelo de componentes.

iv. No que diz respeito ao versionamento dos componentes, considerando que podem coexistir muitos componentes com diferentes implementações, é preciso controlar as versões e os nomes dos componentes para evitar conflitos e possibilitar a substituição dos componentes similares.

v. Em relação à customização dos componentes, deve ser possível modificar as configurações padrões dos componentes, como cores e fontes, por exemplo.

3.3 CONSIDERAÇÕES SOBRE O CAPÍTULO

A arquitetura de referência proposta tem como principal objetivo orientar o desenvolvimento de sistemas colaborativos móveis baseados em componentes. É importante ressaltar que a arquitetura foi definida considerando: aspectos da plataforma móvel, de sistemas colaborativos, e da própria orientação a componente. Neste Capítulo foram apresentados tanto requisitos definidos com base em sistemas colaborativos móveis quanto requisitos definidos com base na orientação a componentes. Além disso foi apresentado um cenário motivador que descreve como é feito o uso dos componentes pelo desenvolvedor de aplicações colaborativas, abordando desde a instalação da infraestrutura e dos componentes, até questões como a atualização de versões, mesmo depois que um usuário já estiver utilizando a aplicação. Esse cenário auxiliou na definição dos requisitos de orientação a componentes.

4 CREAMA

4 CREAMA

Com base nos requisitos descritos no Capítulo anterior foi definida a *Component-Based Reference Architecture for Collaborative Mobile Applications* – CReAMA. Para a definição da arquitetura, também foram considerados aspectos específicos da plataforma Android.

Os trabalhos relacionados mostram uma crescente adoção de arquiteturas orientadas a serviços (*Service Oriented Architecture* – SOA) para usuários móveis (TRUONG e DUSTDAR, 2012). Com isso, CReAMA foi concebida visando o desenvolvimento de aplicações clientes, considerando sistemas colaborativos que funcionem orientados a serviços. Assim, a arquitetura passa a ser utilizada para orientar o desenvolvimento do aplicativo móvel em si, abstraindo os aspectos internos dos serviços a serem acessados pelos aplicativos.

Para facilitar o entendimento, a arquitetura foi dividida em três partes principais: modelo de componentes, sensores e infraestrutura. Cada uma delas representa uma camada da arquitetura () descrita a seguir.

4.1 MODELO DE COMPONENTES

Nesta seção foi abordado o Modelo de Componentes da infraestrutura de CReAMA. Esse modelo se torna importante pois facilita o acoplamento de componentes para que o montador da aplicação consiga, com pouco esforço, realizar o agrupamento de componentes em telas específicas, pois cada tela tem um ou mais componentes associados à elas. Alguns elementos representam, por exemplo, apenas a interface gráfica, como o *MainLayout*, e outros representam a tela a que o componente será acoplado, como é o caso de *CRActivity*.

A Figura 1 apresenta os elementos da arquitetura relacionados à orientação a componentes. Nos diagramas apresentados nesta seção, as classes definidas pela própria plataforma Android são destacadas com um estereótipo (“<<Android>>”). Vale ressaltar que a opção por esta plataforma atende parcialmente os requisitos de “Heterogeneidade” (requisitos *E.i* e *E.ii*), visto que diferentes plataformas de hardware (entre tablets e smartphones) adotam essa tecnologia.

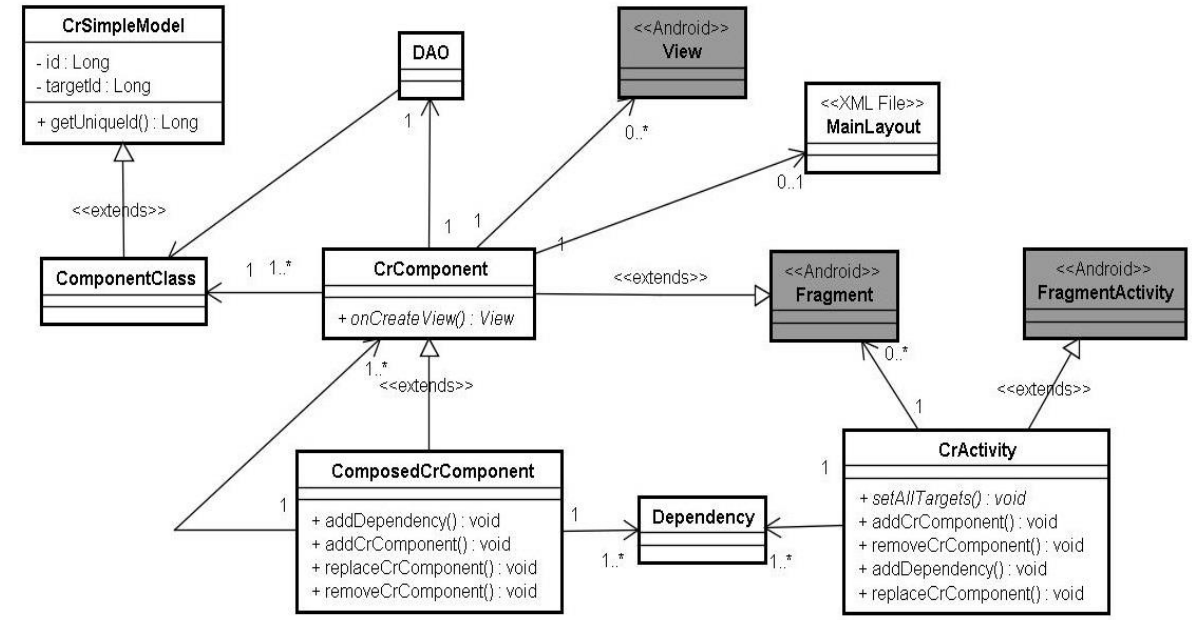


FIGURA 1 - MODELO DE COMPONENTES

Cada tela da aplicação deve ser uma subclasse de *CrActivity* enquanto todos os componentes a serem utilizados são subclasses de *CrComponent*. Por meio da operação *addCrComponent()*, o desenvolvedor decide quais componentes farão parte de cada tela da aplicação. Com isso, a prototipação é facilitada possibilitando que o desenvolvedor monte sua aplicação a partir de componentes já existentes (requisito *J.i*). É importante mencionar também as outras operações, como *replaceCrComponent()* e *removeCrComponent()*, que são utilizadas para substituir e remover componentes, respectivamente.

As três operações descritas possibilitam alterar os componentes que compõe uma *CrAcitivity* específica em tempo de execução. Dessa maneira é possível realizar a prototipação do sistema e realizar testes em relação à interface gráfica compondo os componentes facilmente até que seja atingido um nível satisfatório, ou seja, a arquitetura facilita o esforço dos desenvolvedores no atendimento aos requisitos de “Interface com o Usuário”, possibilitando, por exemplo, que os mesmos construam sistemas com pouco uso de rolagem e com navegação fácil (requisitos *I.i* e *I.ii*).

O agrupamento de componentes (requisito *J.iii*) é possível de duas formas. A primeira opção é utilizar a própria classe *CrActivity*, agrupando-se componentes na mesma tela da aplicação. A segunda é definida com base no padrão de projeto *Composite* (BUSCHMANN et al., 2007), utilizando-se a classe *ComposedCrComponent*. A vantagem, neste caso, é que os componentes compostos representam novos componentes que podem ser reutilizados futuramente (na mesma ou em outras aplicações). Com isso, componentes elementares são subclasses diretas de *CrComponent* enquanto componentes compostos são subclasses diretas de *ComposedCrComponent.*

Os componentes gráficos (*CrComponent* ou *ComposedCrComponent*) estendem a classe *Fragment*. O uso dessa classe possibilita que cada componente seja associado a “parte” de uma tela. Com isso, um ou mais componentes podem ser visualizados dentro de uma mesma tela (*CrActivity*), atendendo-se parcialmente ao requisito de uso de recursos (requisito *H.i*).

É importante mencionar que *CrComposedComponent* e *CrActivity* possuem métodos que agem de forma diferente, mas que possuem o mesmo propósito, que é o de agrupar e desagrupar componentes gráficos na tela ou em um componente composto (*addCrComponent()*, *removeCrComponent()* e *replaceCrComponent()*). Esses métodos agrupam os componentes em suas respectivas telas (*CRActivity*).

Outra maneira de agrupar e desagrupar componentes é configurando dependências entre os componentes com o *addDependency()* – que encontra-se presente nas duas classes. Configurando quais componentes possuem alvos (*targets*), ou seja, se um componente depende do outro, a própria infraestrutura de componentização vai tratar o agrupamento dos componentes, e dessa maneira, existirá um acoplamento lógico entre os componentes, permitindo a comunicação entre eles por meio de retorno de chamadas (*call-backs*) de eventos. Nesse caso, é preciso definir pontos de partida, por exemplo, se o componente ‘C’ e ‘B’ dependem de ‘A’, é preciso acoplar apenas ‘A’ em uma CRActivity e ‘C’ e ‘B’ também serão mostrados de forma recursiva, devido à dependência que foi previamente estabelecida.

Tratando-se da plataforma Android, o *layout* do componente gráfico é definido estaticamente no seu XML. O requisito de customização de componentes (requisito *J.v*) é cumprido pela utilização de APIs nativas do Android, que fornecem

métodos (*getters* e *setters*) para definir e recuperar informações de objetos, possibilitando alterar suas propriedades dinamicamente. Além disso, a implementação do CReAMA é de código aberto, permitindo a customização estática diretamente nos arquivos XML.

Nesta parte da arquitetura, também são definidas classes relacionadas ao padrão de projeto DAO – *Data Acces Object* (BUSCHMANN et al., 2007), usadas para abstrair e encapsular os mecanismos de acesso a dados escondendo os detalhes da execução e origem dos dados. Com isso, é possível oferecer ao desenvolvedor meios de salvar informações de modo persistente para serem acessadas também quando o dispositivo estiver desconectado (*offline*), satisfazendo o requisito de *caching* (requisito *G.i).*

Em relação aos requisitos de desenvolvimento baseado em componentes, para a execução da aplicação, os componentes são baixados junto com a aplicação, encapsulados em um arquivo de extensão APK. Este último inclui arquivos de extensão JAR, que por sua vez, incluem as classes que implementam os serviços dos componentes e seus respectivos arquivos XML utilizados pela plataforma Android.

Para realizar a instalação dos novos componentes em uma aplicação (requisito J.i), é necessário que exista uma definição da aplicação, em um formato padrão, contendo a informação de quais componentes são utilizados por ela, versão da aplicação, data de atualização e a disponibilização desses dados na web (com JSON, por exemplo). Dessa maneira, a aplicação móvel fará a verificação dos dados, e caso já haja uma nova versão da aplicação com diferentes componentes, a reinstalação será feita.

Para a solução do requisito de versionamento (*J.iv)* é proposto que estejam disponíveis no servidor informações sobre a versão dos componentes e da própria aplicação, como citado anteriormente. Dessa maneira, a aplicação terá informação sobre a versão dos componentes que utiliza, e quando for necessário, requisita o download e reinstalação dos mesmos, atualizando assim, os componentes e a aplicação. Ainda em relação ao requisito de versionamento (requisito *J.iv*), foi desenvolvido um diagrama para auxiliar sua implementação. Esse diagrama engloba aspectos relacionados à modificação de componentes e pode ser visto na figura 2.

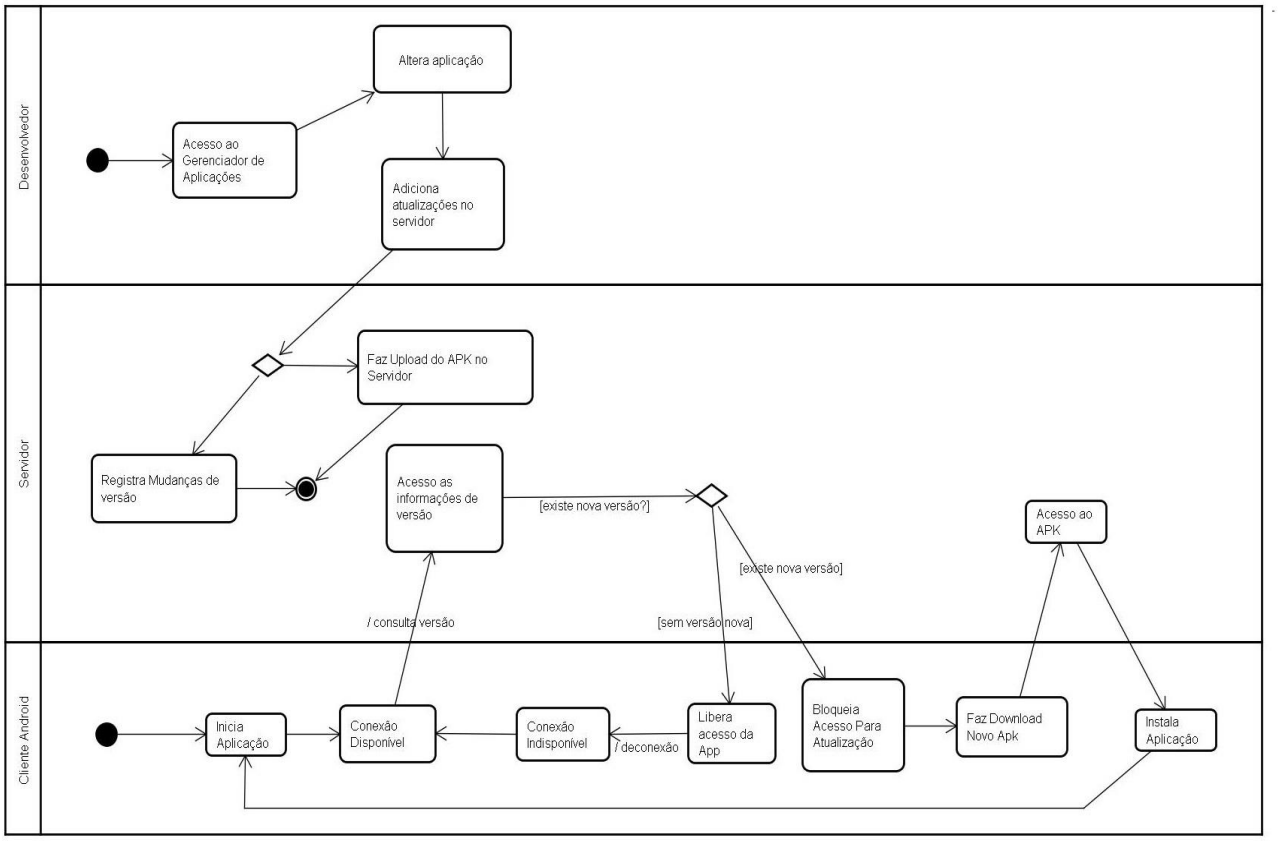


FIGURA 2 - VERSIONAMENTO

A utilização dos componentes (requisito *J.ii*) é algo de extrema importância. Sem detalhes de como usá-los, desenvolvedores podem ter dificuldades que podem gerar muito esforço e perda de tempo através de tentativas de aprender sobre os componentes sem nenhum auxílio de documentação. Para contornar isso, os componentes devem estar catalogados e documentados com exemplos, diagramas e textos de fácil entendimento.

É necessário que os componentes estejam disponíveis (requisito *J.vi*) em um repositório definido, assim como as informações de suas respectivas versões e um documento com tudo que seja necessário para saber como desenvolvedores devem utilizá-los. O documento deve incluir um catálogo contendo detalhes de todos os componentes e arquitetura propostos. É importante mencionar que os requisitos *J.ii* e *J.vi* servem como orientação de trabalho no ambiente organizacional em que os desenvolvedores estiverem.

4.2 SENSORES

A Figura 3 apresenta os elementos da arquitetura relacionados ao acesso a sensores da plataforma. Essa parte da arquitetura foi definida visando-se cumprir os requisitos relacionados ao uso de informações de rede, bateria, localização, etc.

Os componentes têm acesso aos sensores por meio de um serviço local denominado *SensorsService*. A definição de um serviço local visa oferecer uma interface comum de acesso aos diferentes sensores da plataforma. Esse serviço principal é usado para obter informações sobre o próprio dispositivo para que as informações possam ser, por exemplo, exibidas ao usuário ou enviadas via rede aos serviços do sistema colaborativo. O sistema pode, com isso, obter informações de contexto do usuário, ou implementar serviços que combinem essas informações.

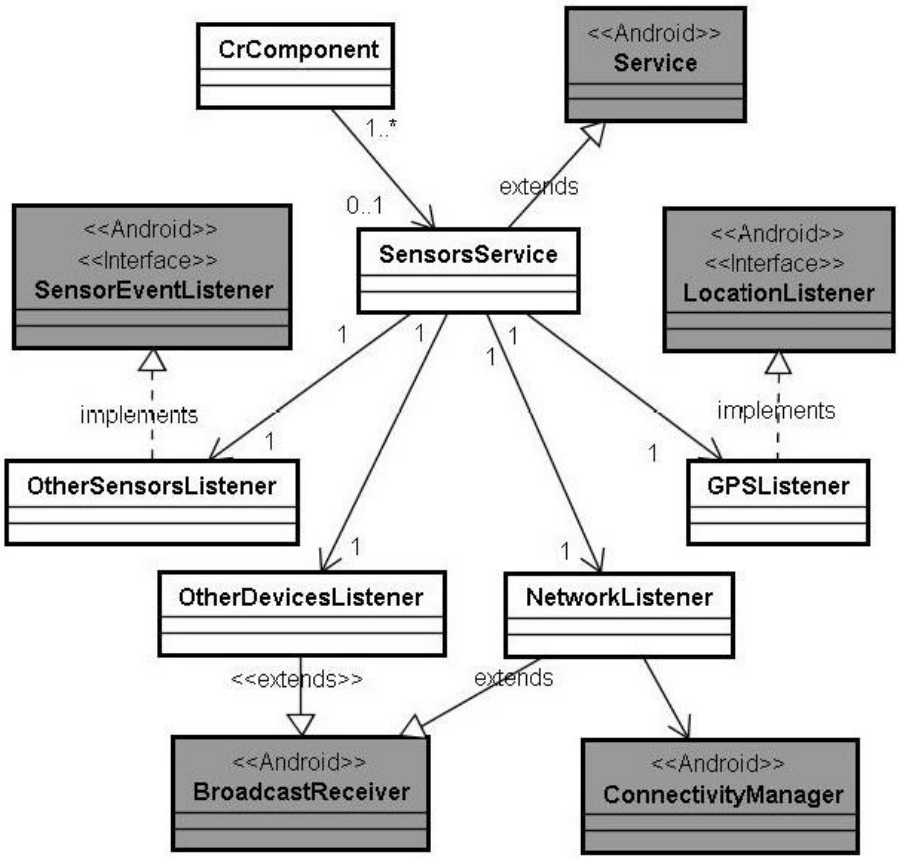


FIGURA 3 - SENSORES

O uso de um serviço local que oferece uma interface comum para acessar os diferentes sensores da plataforma é vantajoso devido à maneira na qual as APIs do Android implementam o uso de tais recursos. Giroscópio, acelerômetro e sensor

de luz podem ser acessados por meio da implementação de uma interface (*SensorEventListener*) diferente da qual os dados do GPS (*LocationListener*) podem ser acessados. Além disso, a maneira de obter dados de bateria e memória externa, por exemplo, diferem das duas maneiras de acesso dos sensores citados anteriormente. Como no Android o acesso é feito de maneira diferente através das APIs, na arquitetura foi necessária a criação da classe "*OtherDevicesListener*".

A classe *NetworkListener* é necessária para fornecer a percepção ao usuário sobre a conectividade do dispositivo, por isso ela utiliza a classe do Android ConnectivityManager, que fornece dados de qual rede o usuário está conectado, por exemplo. Essa classe faz um papel ativo na infraestrutura, ou seja, ela avisa automaticamente sempre que um usuário perde a conectividade ou se a conexão está ativa em um determinado momento.

Utilizando-se os elementos descritos, desenvolvedores podem construir componentes específicos para suas aplicações que satisfaçam requisitos que não são cumpridos diretamente por CReAMA, como por exemplo, uma lista de usuários conectados (requisito *A.i*). Nesse caso, consideremos um cenário onde o cliente móvel faça o envio de tempos em tempos avisando se está conectado, ficando a cargo do servidor salvar essas informações e fornecer as informações de quais usuários estão conectados por meio de um serviço. O componente Lista de Usuários Conectados faria a requisição para esse serviço e mostraria quais usuários estão disponíveis no momento.

Com essas informações é igualmente possível desenvolver componentes para satisfazer os requisitos categorizados em "Awareness" (requisitos *F.i*, *F.ii* e *F.iii*) e "Autonomia de Aplicação" (requisitos *B.i*, *B.ii* e *B.iii*).

4.3 INFRAESTRUTURA

Finalmente, a Figura 4 apresenta os elementos da arquitetura relacionados à infraestrutura. Essa parte da arquitetura foi definida para gerenciar o acesso por parte dos componentes da aplicação ao sistema colaborativo via rede.

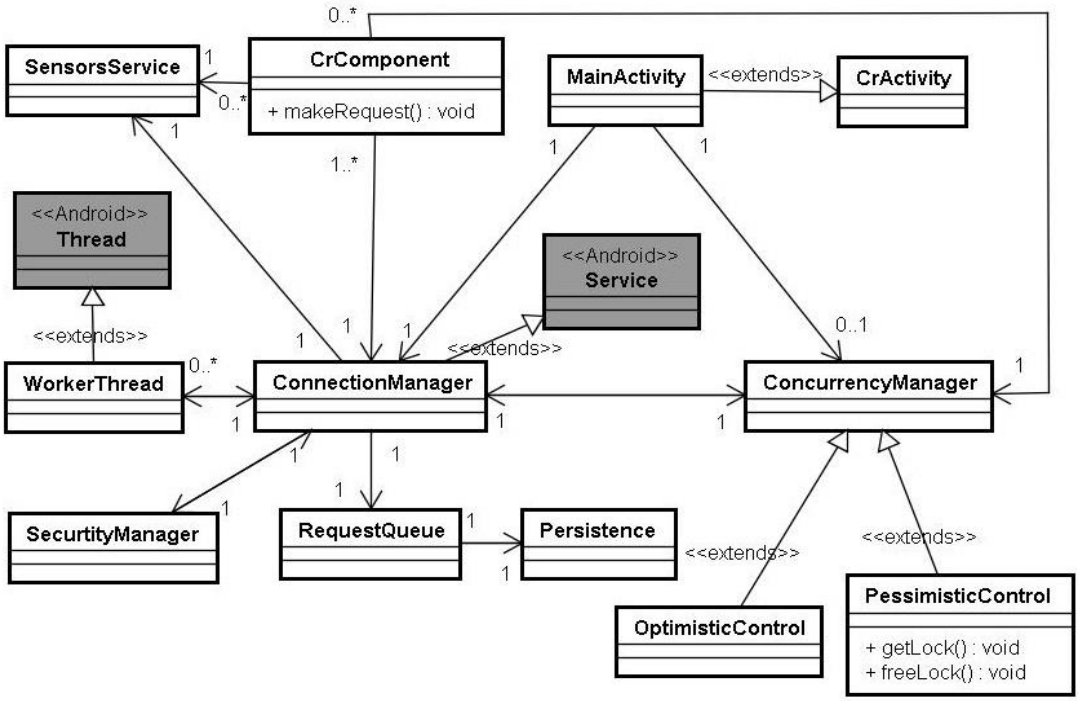


FIGURA 4 - INFRAESTRUTURA

A comunicação com o sistema colaborativo se faz por meio de chamadas realizadas pelos componentes aos serviços do sistema. Essas chamadas são tratadas como requisições pelo *ConnectionManager*. As requisições são atendidas conforme o escalonamento definido pela *RequestQueue*. Para cada requisição é criada uma *WorkerThread*, responsável por gerenciar essa requisição.

Face às limitações de rede (requisito *H.ii*), um papel importante do *ConnectionManager* é gerenciar a intermitência no canal de conexão sem fio, visando garantir que cada *WorkerThread* consiga completar sua requisição. Se os usuários estiverem trabalhando *offline*, as requisições permanecem na *RequestQueue*. Se o usuário estiver *online* e quiser colaborar, as requisições na fila são atendidas, permitindo ao usuário escolher seu nível de participação nas atividades, cumprindo o requisito de colaboração sob demanda (requisito *A.ii*).

Ressalta-se que para fazer o controle do estado *offline* e *online*, o usuário pode acionar os dispositivos de conexão diretamente no próprio Android, ou por meio de um controle que é oferecido pelo *ConnectionManager*. Nesse último caso, apenas a aplicação não acessará a rede. O *ConnectionManager*, com base em

informações sobre a bateria, pode sugerir ao usuário que entre em modo *offline*. Com isso, atende-se em parte o requisito de uso de recursos (requisito *H.i*).

A classe *SecurtityManager* deve ser utilizada pelo *ConnectionManager* caso a aplicação faça uso de mecanismos de segurança, como protocolos seguros ou chaves de segurança. Com isso, os controles de segurança e privacidade definidos nos requisitos de “Proteção” (requisitos *C.i*, *C.ii*, *C.iii*), em parte podem ser atendidos por componentes específicos para realização de *login* e controle de usuário, e em parte pelo *SecurityManager*.

No que diz respeito ao requisito de resolução de conflitos (requisito *G.ii*), os usuários móveis podem atualizar as informações locais sobre a aplicação quando estão trabalhando sozinhos (*offline*). Eventualmente, isso pode gerar inconsistências nos dados compartilhados. A sincronização de dados requer algoritmos de resolução de conflitos para conciliar essa informação, a fim de manter uma visão comum do ambiente compartilhado, sem inconsistências.

Dessa maneira, foram definidas as classes *OptimisticControl* e *PessimisticControl* para implementar o controle de concorrência otimista e pessimista (BORGHOFF e SCHILICHTER, 2000) no acesso concorrente às informações compartilhadas. No controle otimista, de maneira geral, o servidor é responsável por gerenciar a concorrência, enquanto é permitido que o cliente modifique os dados livremente. Já no caso do controle pessimista, o cliente precisa pedir permissão para alterar determinados recursos, enquanto o servidor é responsável por gerenciar qual cliente tem a permissão de acesso dos dados. Nesse caso, foram definidos métodos básicos para tratar o controle pessimista: *getLock()* e *freeLock()* e se a aplicação fizer uso de controle pessimista será necessário implementar esses métodos. Caso o desenvolvedor utilize outro mecanismo de concorrência, basta estender a classe e implementar as operações necessárias.

É importante mencionar que se houver algum tipo de controle de concorrência, as classes citadas devem ser utilizadas pelos componentes da aplicação que necessitem desse tipo de controle. A comunicação dos componentes que utilizam as classes de controle de concorrência deve ser sempre intermediada pelo *ConcurrencyManager*. Este último repassa o controle para a classe de controle da aplicação, ou seja, o próprio CRComponent, que vai ser referente ao componente específico.

Vale mencionar que muitos requisitos levantados na seção 3.2 são dependentes do tipo de aplicação. Exemplos que não foram explicitamente citados são os requisitos de “Comunicação” (requisitos *D.i* a *D.iv*) e o requisito de capacidade de gerenciamento (requisito *I.iii*). O importante neste caso é que a arquitetura possa prover meios de se obter as informações necessárias para que a aplicação em si atenda aos requisitos implementando componentes, como foi citado no caso de “Awareness” e “Autonomia de Aplicação”.

Para atender aos requisitos que são dependentes do tipo de aplicação, foram propostos serviços em nível de Infraestrutura (CReAMA), como já foi citado anteriormente. Serviços são usados para acessar informações do próprio dispositivo, fornecendo ao servidor as informações de contexto do usuário, como por exemplo, se determinados usuários estão disponíveis ou qual a localização geográfica de um determinado usuário. Dessa maneira, é possível construir componentes como lista de usuários conectados, lista dos últimos lugares que um usuário esteve, entre outras funcionalidades.

4.4 CONSIDERAÇÕES SOBRE O CAPÍTULO

Neste Capítulo foi apresentada CReAMA, uma arquitetura de referência para orientar o desenvolvimento de sistemas colaborativos móveis orientados a componentes. Ressalta-se que CReAMA foi concebida visando o desenvolvimento de aplicações clientes, considerando um sistema colaborativo que trabalhe orientado a serviços. Com isso, a arquitetura de referência deve ser utilizada para orientar o desenvolvimento do aplicativo móvel em si, abstraindo os aspectos internos dos serviços a serem acessados pelos aplicativos. Todos os requisitos apresentados no Capítulo anterior foram mapeados para os diagramas ou são descritos como orientações no ambiente de trabalho, e foram mapeados a uma das três partes da arquitetura de referência: modelo de componentes, sensores e infraestrutura.

5 IMPLEMENTAÇÃO E TESTES

5 IMPLEMENTAÇÃO E TESTES

Para avaliar a arquitetura de referência, primeiramente foi realizada uma instanciação da arquitetura, denominada CReAMA Tools. A partir desse ferramental foi desenvolvido o primeiro conjunto de componentes, denominado GW-Android (Melotti e Gomes, 2012), que representa uma extensão do Groupware Workbench (GW). Utilizando-se esses componentes, os dois primeiro protótipos de aplicação foram desenvolvidos no contexto do GW: um sistema de perguntas e respostas e a versão móvel da rede social Arquigrafia (Michalsky, 2012).

Na sequência, foi realizada uma análise da arquitetura desenvolvida com base nos requisitos definidos anteriormente para verificar se os mesmos foram atendidos. Depois do desenvolvimento dos dois primeiros protótipos que serviram como prova de conceito, foi realizado um estudo de caso que consiste em uma comparação na utilização e não utilização de CReAMA no desenvolvimento de duas aplicações do sistema colaborativo Conecte Ideias.

De uma forma geral, para atender o objetivo de analisar a proposta de arquitetura de referência apresentada neste trabalho, este Capítulo descreve os três procedimentos metodológicos aplicados:

1. A partir da execução de uma instância denominada CReAMA Tools, que foi implementada com base na arquitetura de referência CReAMA, foram desenvolvidos dois conjuntos de componentes (*ComponentKits* ou *ToolKits*) e três aplicações que utilizam estes *ToolKits* com o intuito de apresentar sua aplicabilidade e cumprimento dos requisitos definidos anteriormente.
2. Foram efetuados testes baseados em um estudo de caso selecionado que consiste em uma comparação na utilização e não utilização de CReAMA no desenvolvimento de duas aplicações do sistema colaborativo Conecte Ideias. A comparação foi feita de duas maneiras: a primeira consiste em utilizar métricas de software para comparar os dois projetos apontando benefícios e implicações para cada métrica; a segunda consiste em analisar

características de suporte à mobilidade em cada uma das aplicações, apontando qual sistema possui maiores vantagens e benefícios.

3. Publicação de artigos em eventos científicos para avaliação da proposta durante todo o percurso do trabalho. Este processo foi de essencial importância para a evolução do trabalho, com sugestões, questões a serem melhor definidas e críticas para redefinições dos rumos do trabalho durante todo o seu ciclo de definição, modelagem e projeto.

Para melhorar a compreensão e entendimento em relação a ferramentas e protótipos que foram implementados, é apresentado um diagrama que mostra, entre outros aspectos, como os protótipos se relacionam com os seus respectivos *ToolKits* e a infraestrutura CReAMA Tools. CReAMA Tools é a base e a parte comum de todo ferramental desenvolvido e pode ser utilizado na construção de futuros *Toolkits*. O diagrama é apresentado na Figura 5:

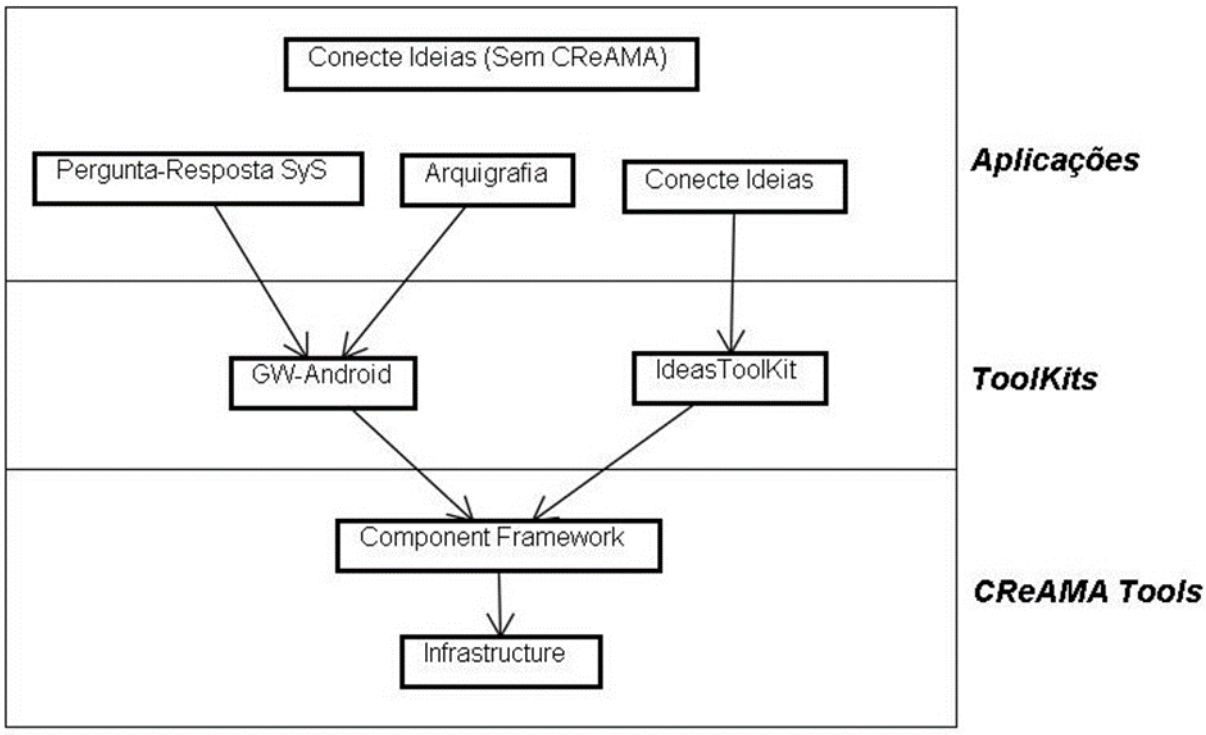


FIGURA 5 - DESENVOLVIMENTO COMPUTACIONAL

5.1 CREAMA-TOOLS

Desenvolvido com base na arquitetura de referência, o CReAMA Tools é composto por duas partes principais: Infraestrutura (incluindo o serviço de sensores) e Framework Component.

A Infraestrutura consiste em um conjunto de serviços locais que são responsáveis por gerenciar, entre outras coisas, a comunicação, conexão, persistência e sensores. O Component Framework consiste em um conjunto de funcionalidades relacionadas aos componentes responsáveis por gerenciar, inicializar, executar, agrupar e desativar os componentes, facilitando a prototipação de aplicações colaborativas móveis.

Alguns serviços desenvolvidos são usados para acessar informações do próprio dispositivo, fornecendo ao servidor as informações de contexto do usuário, como atividades desenvolvidas, última modificação em algum documento, se o usuário está disponível, entre outras informações (MELOTTI et al., 2013). Apesar do estudo de caso não abordar todos esses aspectos, com esse conjunto de serviços será possível construir componentes como lista de usuários conectados, lista de documentos alterados no dia, lista de atividades desenvolvidas por usuário de um mesmo grupo de trabalho, etc. Todos esses serviços foram desenvolvidos com base na arquitetura de referência proposta.

Vale mencionar que muitos requisitos levantados no Capítulo 3 são dependentes do tipo de aplicação. Exemplos que não foram explicitamente citados são os requisitos de “Comunicação” (requisitos *D.i* a *D.iv*) e o requisito de capacidade de gerenciamento (requisito *I.iii*). O importante nesse caso é que a arquitetura proveja meios de se obter as informações necessárias para que a aplicação em si atenda aos requisitos implementando componentes, como foi citado previamente no caso de “Awareness” e “Autonomia de Aplicação” (MELOTTI et al., 2013).

Depois da fase de desenvolvimento, foi realizada uma análise de CReAMA Tools com base nos requisitos definidos anteriormente na arquitetura de referência. Para cada requisito foi feito o questionamento: A implementação da arquitetura atende ao requisito ‘x’? Como todos os requisitos foram atendidos pelo menos parcialmente, as possíveis respostas foram: atende diretamente (AD), provê meios de atender (PMA) e atende parcialmente (AP). O resultado dessa análise é apresentado na Tabela 1 no final deste Capítulo.

5.2 GW-ANDROID

O GW Android foi desenvolvido como uma extensão do modelo de componentes do Groupware Workbench para a plataforma móvel Android. Representa um ferramental para o desenvolvimento de sistemas colaborativos móveis no domínio do GW, em que o desenvolvedor não precisa conhecer os detalhes de implementação dos componentes para utilizá-los e criar novos sistemas colaborativos. Ressalta-se que uma limitação do GW era o fato deste não oferecer suporte a clientes móveis para os sistemas colaborativos desenvolvidos com base nele.

Na abordagem da plataforma GW, os componentes são baseados no modelo 3C de colaboração, em que colaboração é analisada a partir das dimensões de comunicação, coordenação e cooperação (GEROSA e FUKS, 2008). A tecnologia utilizada no GW oferece suporte ao padrão MVC, à propagação de eventos, à persistência de dados e à criação de elementos de interface. Os componentes chamados de *collablets* são utilizados de forma hierárquica para construir as ferramentas de colaboração (MAMANI e GEROSA, 2011).

O GW-Android deve ser responsável pela integração entre as tecnologias Web 2.0 utilizadas pelo GW e as tecnologias da plataforma móvel. Ele deve prover acesso eficiente aos recursos do dispositivo móvel no que diz respeito ao uso de energia, assim como gerenciar a desconectividade temporária de usuários devido a instabilidades das conexões sem fio.

Além do desenvolvimento do conjunto de componentes, denominado GW-Android, torna-se também importante a definição de clientes móveis que representem os ambientes de execução das aplicações do Groupware-Workbench. No caso do presente trabalho, duas aplicações foram selecionadas: *Sistema de Perguntas e Respostas* e *Arquigrafia*. O desenvolvimento desses protótipos teve como objetivo demonstrar a aplicabilidade do GW-Android.

No desenvolvimento do GW-Android, foram primeiramente implementados os componentes elementares (que não são compostos de outros componentes). Dentre os componentes implementados encontram-se o *Comment*, *Photo*, *Tag*, *Faq*, *Georeference*, *Rating*, *Binomio*, *e User*. Utilizando-se esses componentes, foram criados componentes compostos com base no padrão Composite. Como exemplo desse tipo de composição, podemos citar um componente *GeoRefComment*, criado a partir de um componente *Georeference* e um componente *Comment*.

A partir dos componentes elementares foram derivados os componentes gráficos, que são utilizados para montar a aplicação. Esses componentes refletem ações que um usuário pode executar dentro do sistema, como visualizar uma foto e executar o *login*. Dentro do GW-Android, os componentes gráficos desenvolvidos foram: *PhotoView*, *PhotoGalleryView*, *PhotoUpload*, *FaqView*, *FaqEdit*, *FaqSend*, *FaqListView*, *LoginExecute*, *UserProfileView*, *BinomioSend*, *BinomioAverageView*, *PhotoTrackerMap*, *UserTrackerMap*, *CommentSend*, *CommentListView*, CommentView, *TagView*, e *TagListView*. A seguir está um diagrama relacionando os componentes gráficos com os seus respectivos elementos:

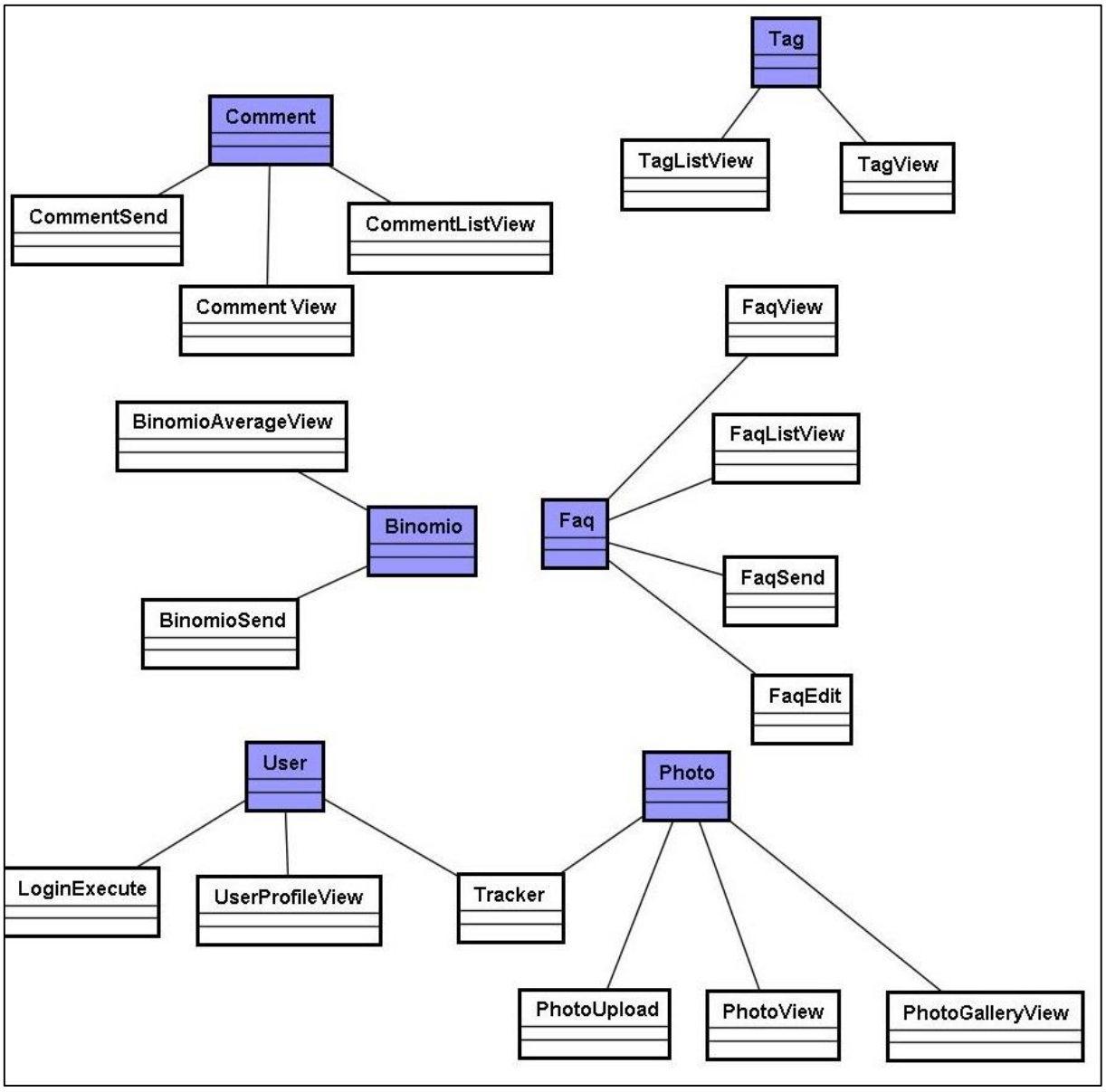


FIGURA 6 – COMPONENTES DO GW-ANDROID

A implementação dos dois primeiros protótipos utilizam os componentes do GW-Android, que por sua vez, utiliza o CReAMA Tools. Então os protótipos e o GW-Android utilizam as duas partes principais de CReAMA Tools: (i) a infraestrutura (sensores foram incluídos) e (ii) o ComponentFramework para apoiar a execução dos componentes. No primeiro protótipo (Sistema de Perguntas e Respostas), os componentes implementados são uma extensão de componentes já definidos pelo GW, e a aplicação desenvolvida pode ser utilizada por meio de qualquer aparelho com a plataforma Android.

No segundo protótipo, além dos componentes estendidos do GW, também foram criados três componentes não presentes originalmente no GW: *PhotoTrackerMap*, *UserTrackerMap*, e *GeoRefComment*. O *PhotoTrackerMap* e o *UserTrackerMap* seriam usados junto com mapas para mostrar a localização geográfica de usuários disponíveis e fotos cujos locais poderiam também ser adicionados aos mapas referenciados através de coordenadas geográficas. No entanto, após o desenvolvimento e teste (preliminares) desses componentes, houve mudanças na API Android dos mapas utilizados para implementar os componentes citados. Por limitações de tempo, não foi possível portar esses componentes para Versão 2 do GoogleMapsApi. Com isso, os mesmos foram retirados da versão atual do protótipo, ficando essa tarefa como trabalho futuro. Quanto ao componente *GeoRefComment*, ele foi desenvolvido apenas para testar a composição de componentes utilizando a classe *CRComposedComponent*.

5.2.1 Sistemas de Perguntas e Respostas

O sistema de perguntas e respostas desenvolvido pode ser utilizado de forma colaborativa para alterar as perguntas e respostas de domínios específicos. Um exemplo para a utilização desse sistema seria para a construção de conhecimento de forma colaborativa com professores e alunos de alguma matéria, por exemplo. No contexto dessa dissertação, as perguntas e respostas foram relacionadas ao Groupware-Workbench e CReAMA.

Os requisitos levantados para essa aplicação são apresentados a seguir no sob a forma de um diagrama de casos de uso (Figura 7):

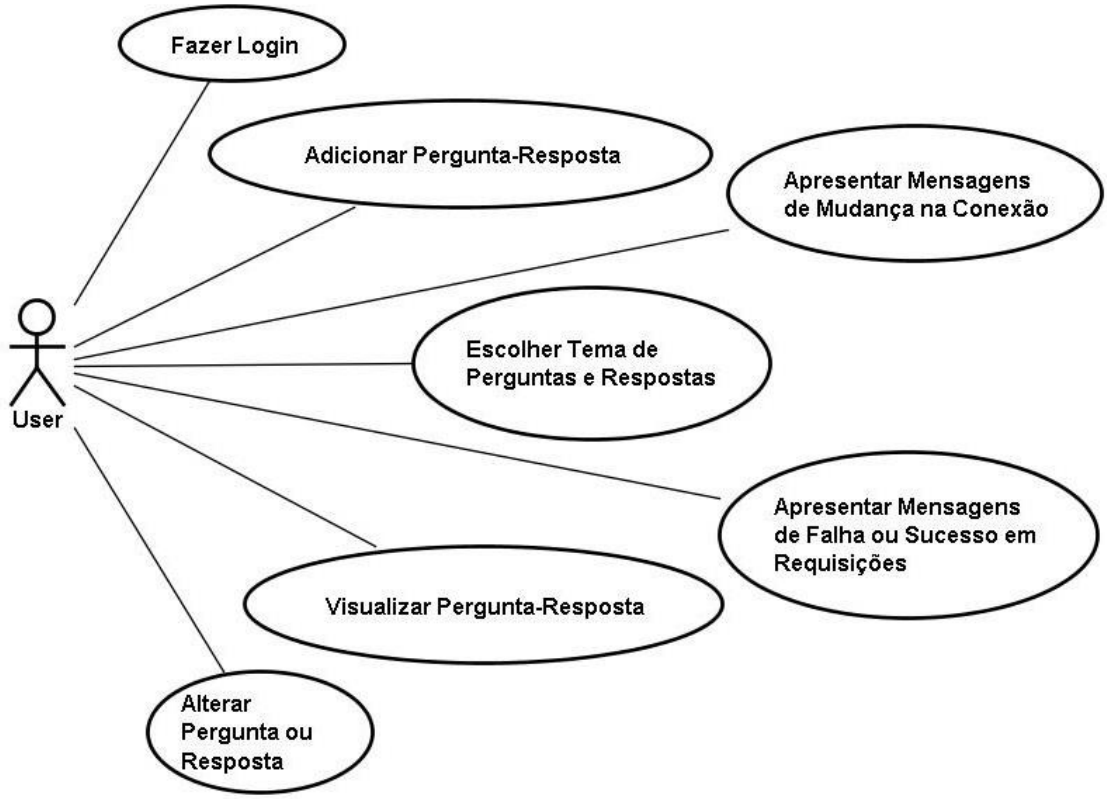


FIGURA 7 - CASOS DE USO DO SISTEMA DE PERGUNTAS E RESPOSTAS

- Execução do login na plataforma de perguntas e respostas;
- Adição de perguntas e respostas;
- Possibilidade da escolha de tema de um domínio específico;
- Visualização de perguntas e respostas de sobre um determinado assunto;
- Alteração, de perguntas e respostas;
- Possibilidade de trabalho *offline* em que o usuário possa realizar todas as ações acima sem ter acesso à Internet para quando a conexão estiver disponível, seja feita a postagem dos dados criados/alterados enquanto *offline*;
- Apresentação de mensagens ao usuário quando houver mudanças no estado de conexão (conectado e desconectado).
- Apresentação de mensagens ao usuário quando as requisições forem processadas pelo serviço (exemplo: listagem recebida com sucesso).

O sistema de perguntas e resposta móvel foi denominado “Pergunta-Resposta SyS”. Este cliente deve se conectar ao serviço de Faq originalmente implementado com

base no GW. Com isso, o acesso ao sistema de perguntas e respostas se faz por meio de uma interface *Rest*, permitindo que a aplicação móvel recupere os dados de determinadas perguntas ou respostas fazendo uma requisição de uma Url específica. Para a realização dos testes esse serviço foi primeiramente configurado (por meio de uma aplicação web) para possuir um estado inicial.

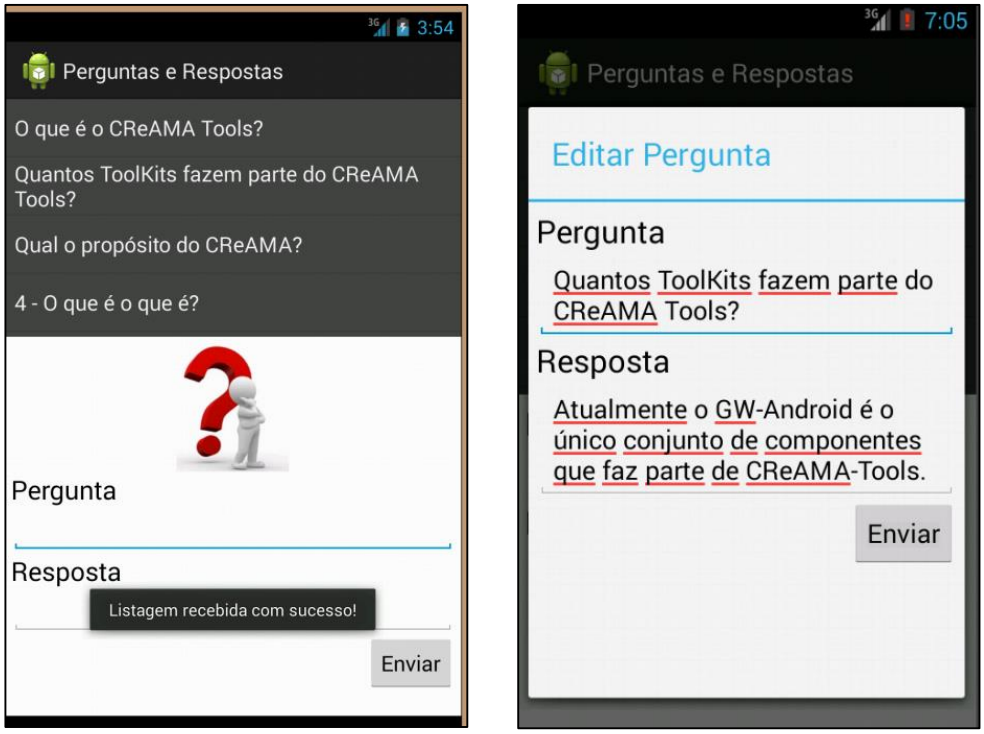
(A) (B)

FIGURA 8: (A) TELA INICIAL DO PERGUNTA-RESPOSTA SYS; (B) ESCOLHA DE TEMAS

A Figura 8 ilustra as telas iniciais do Pergunta-Resposta SyS: a tela antes do login e a tela exibida uma vez que o usuário se conecta. Na tela apresentada na Figura 8 (a), o único componente utilizado é o *LoginExecute*, que é apresentado em uma nova janela depois do click do usuário no botão “Faça o Login”. Depois de feito login, a requisição é verificada pelo serviço Web, que retorna em seguida os dados para que o sistema apresente as possibilidades de temas para visualização de perguntas e respostas (Figura 8b).

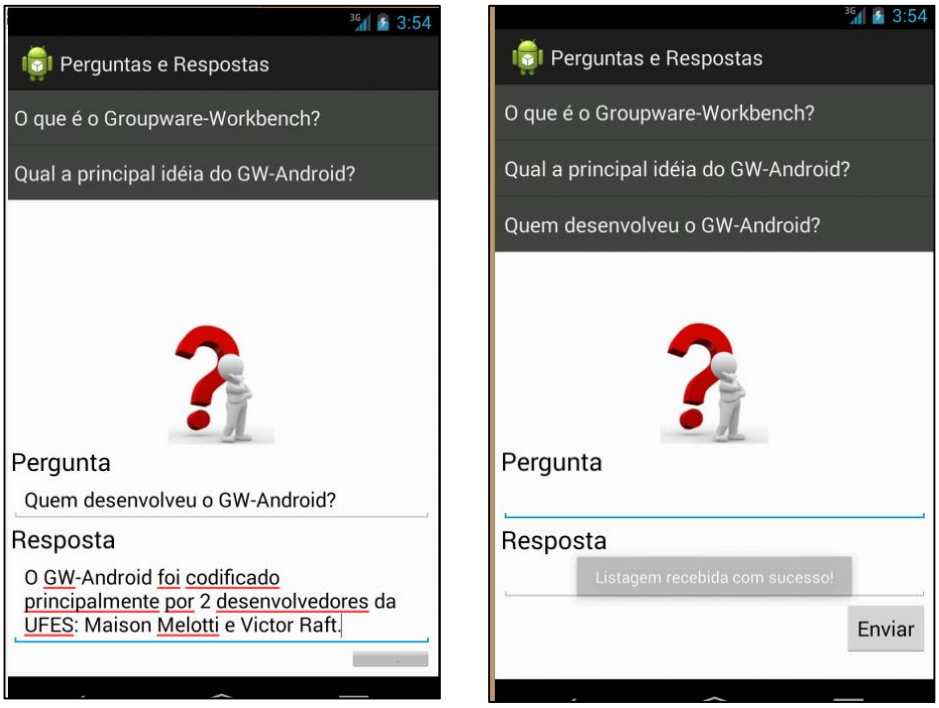
Uma vez selecionado o tema, a tela carrega as perguntas relacionadas (Figura 9a). Nessa tela, os componentes presentes são dois: “FaqListView” e “FaqSend”. A tela também exibe uma mensagem indicando que os dados foram carregados com sucesso (caso contrário, seria exibida uma mensagem de erro). Da mesma maneira, se o usuário perder a conexão de dados com a Internet, é mostrada a mensagem de conexão indisponível.

A tela apresentada pela Figura 9b permite a alteração de uma pergunta ou resposta. Nessa tela é apresentado o componente *FaqEdit*. Uma vez realizada a alteração da resposta, os dados são enviados ao serviço.



(A) (B)

FIGURA 9: (A) LISTA DE PERGUNTAS E RESPOSTAS; (B) COMPONENTE FAQ EDIT



(A) (B)

FIGURA 10: (A) COMPONENTE FAQ SEND; (B) CONFIRMAÇÃO DO ENVIO DA PERGUNTA

A Figura 10a ilustra a tela usada para adicionar uma nova pergunta e resposta. Nessa tela estão presentes os componentes *FaqListView* e *Faq*. Eles ficam ativos na tela, mesmo que o foco seja apenas um deles. Logo depois do envio da nova pergunta e resposta, a página é atualizada. A tela é carregada automaticamente e o componente *FaqListView* é atualizado com uma nova pergunta demonstrando que foi efetuado com sucesso a adição da nova pergunta e resposta (Figura 10b).

5.2.2 Arquigrafia Mobile

O projeto da Rede Social Arquigrafia reúne desde 2009 uma equipe multidisciplinar de pesquisadores da FAUUSP, IMEUSP e ECAUSP para a criação de um ambiente colaborativo para a visualização, interação e compartilhamento de imagens digitais de arquitetura – fotografias, desenhos e vídeos – na Internet. O objetivo principal do projeto, além de contribuir para o estudo, docência, pesquisa e difusão da cultura

arquitetônica, é promover interações colaborativas entre pessoas e instituições. (MICHALSKY, 2012).

A Figura 11 apresenta o diagrama de casos de uso do sistema Arquigrafia, o qual representa uma rede social baseada em imagens da arquitetura brasileira. Nesta seção são mostradas as telas com funcionalidades da aplicação móvel “Arquigrafia Mobile” que refletem as mesmas funcionalidades do sistema Arquigrafia na Web.

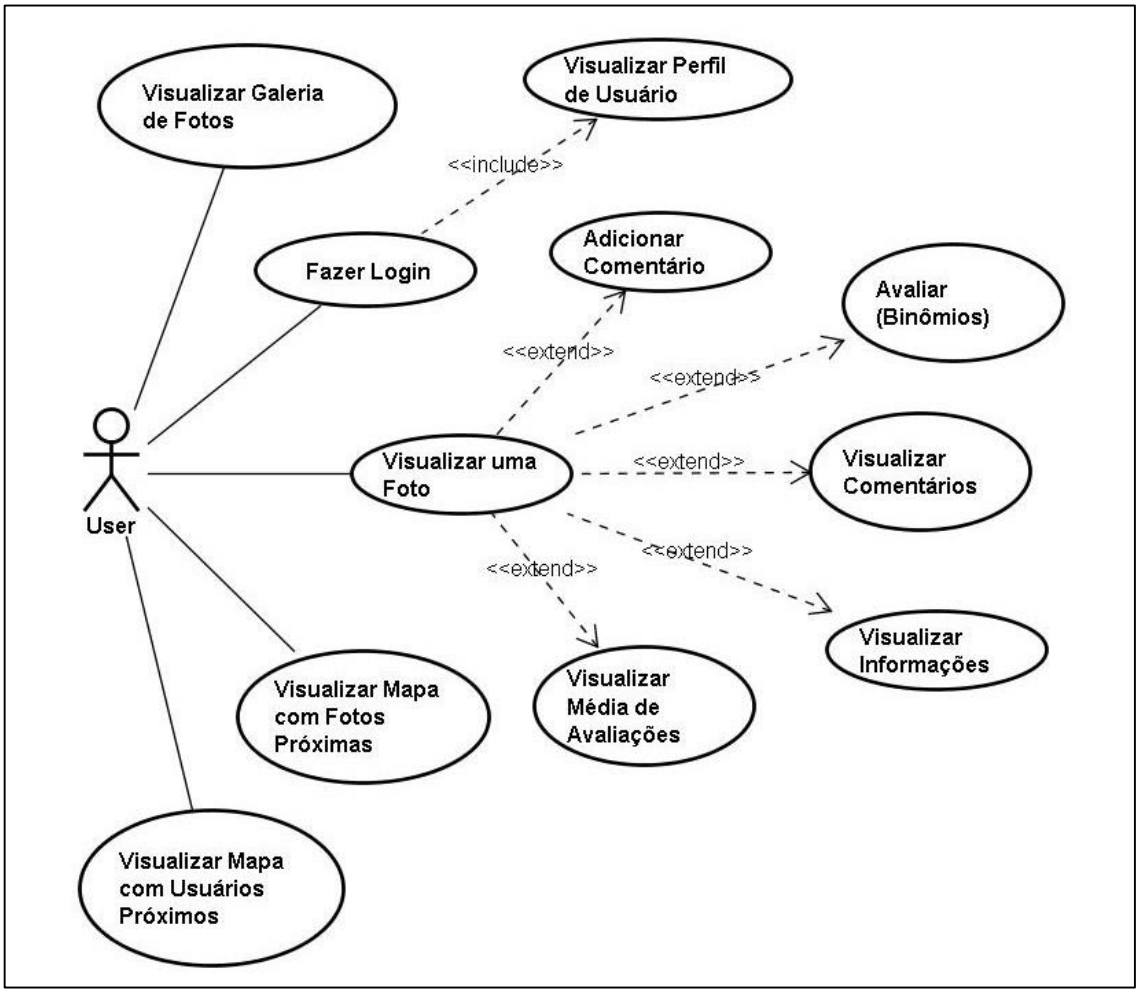


FIGURA 8 - CASOS DE USO ARQUIGRAFIA

A Figura 12 exibe a tela de login do Arquigrafia Mobile, a qual provê informações (percepção) sobre o estado da conectividade do dispositivo. Para efetivação do login, é feita uma verificação junto ao sistema Arquigrafia Web, e, em caso positivo, o usuário também verá uma mensagem confirmando o sucesso.

FIGURA 92 - LOGIN ARQUIGRAFIA

FIGURA 103 – PÓS LOGIN ARQUIGRAFIA

Na sequência, o usuário entra na tela (*CRActivity*) em que é exibida a galeria de fotos do Arquigrafia, implementada por meio dos componentes *PhotoGalleryView* e *UserProfileView* (Figura 13). Nota-se que no sistema mobile, a aparência da galeria é bem diferente da versão Web (Figura 14). Optou-se por um sistema de galeria que mostre as fotos apenas na horizontal podendo ser passadas da esquerda para direita e vice-versa.

Na versão Mobile, a *CRActivity* da Figura 13 também apresenta o componente *UserProfileView*, utilizado para exibir informações básicas do usuário que está conectado. Além disso, há também os botões posicionados abaixo desse componente, utilizados carregar outras duas *CRActivity*s contendo os componentes *PhotoTrackerMap* e *UserTrackerMap*, respectivamente.

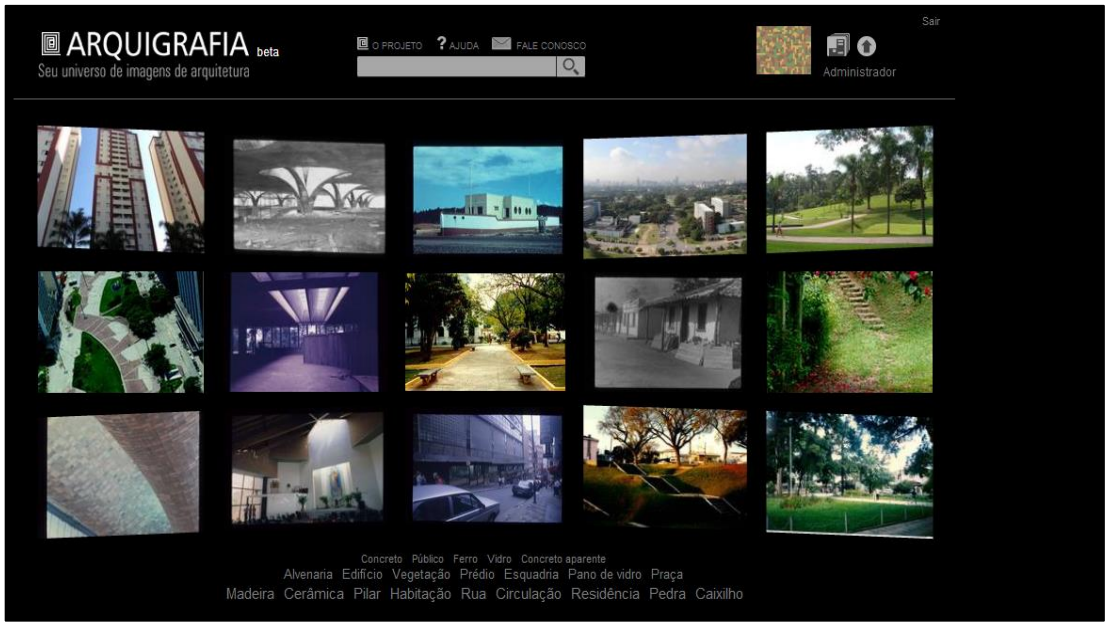


FIGURA 114 - GALERIA NA VERSÃO WEB

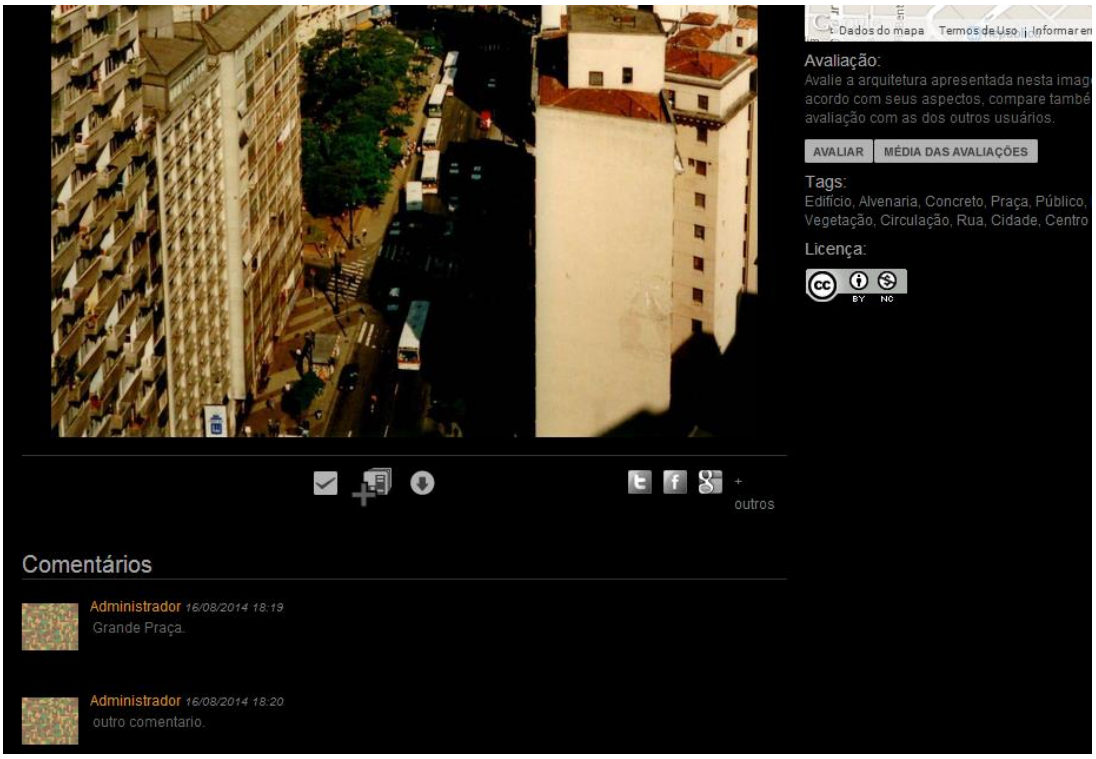


FIGURA 125 – VISUALIZAÇÃO DE FOTO

(A) (B)

FIGURA 16: (A) TAGS NO ARQUIGRAFIA; (B) COMENTÁRIOS NO ARQUIGRAFIA

Na versão Web (Figura 15), ao se visualizar uma foto, também são exibidos os comentários e tags relacionados a ela. A Figura 16a apresenta a *CRActivity* utilizada visualizar uma foto na versão mobile (utilizando-se o componente *PhotoView*). Esta *CRActivity* contém três componentes acoplados a ela: *PhotoGalleryView*, *PhotoView* e *TagListView*. Nem todos os componentes são exibidos devido ao tamanho da tela, o *PhotoGalleryView*. Observe que os comentários e avaliações só podem ser visualizados/realizados a partir de *clicks* nos botões apresentados (isso foi feito para poupar espaço na tela do dispositivo móvel e evitar muita rolagem).

A Figura 16b exibe a *CRActivity* utilizada para visualizar comentários e comentar uma foto. Essa *CRActivity* contém três componentes acoplados a ela: *PhotoView*, *CommentListView* e *CommentSend*. Por exemplo, o comentário “outro comentário”, exibido no canto inferior esquerdo da Figura 15 foi enviado utilizando-se essa *CRActivity*.

A Figura 17 exibe a *CRActivity* utilizada para avaliar uma foto. As avaliações no Arquigrafia são feitas por meio de binômios. Então dois componentes foram criados para representar o binômio: *BinomioSend* e *BinomioAverageView*. Na figura, abaixo além do componente PhotoView, a *CRActivity* também conta com o componente BinomioSend, que permite que as avaliações sejam enviadas para o servidor. Na Figura 18 é apresentada parte da tela do navegador web que exibe o binômio.

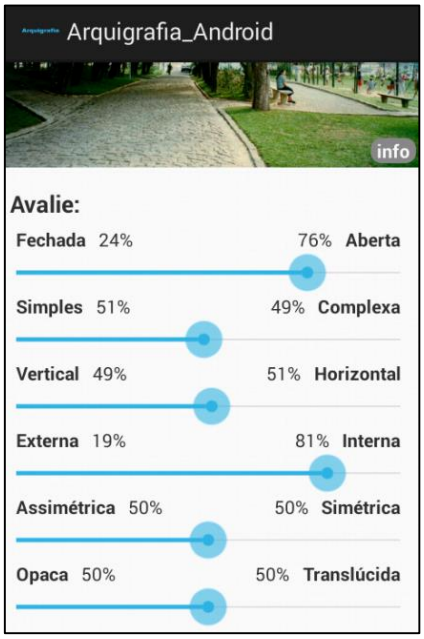


FIGURA 137 – BINOMIOS NO ARQUIGRAFIA MOBILE

FIGURA 18 - BINOMIOS NO ARQUIGRAFIA WEB

5.2.3 Testes

Os testes dos protótipos do Pergunta-Reposta SyS e do Arquigrafia Mobile foram realizados com o intuito de verificar se as funcionalidades previstas foram corretamente implementadas. Particularmente, visava-se verificar se os componentes desenvolvidos para a aplicação móvel respondiam de forma equivalente aos componentes previamente existentes nas aplicações web.

Primeiramente, durante a fase de implementação dos componentes foram realizados testes de unidade em cada componente implementado. Para cada componente, foram realizados em seguida testes de integração, visando verificar se a comunicação entre os componentes e os demais elementos da arquitetura (GW-Android) ocorria corretamente. Por fim, uma vez integrados todos os componentes,

foram realizados teste funcionais, utilizando como base os casos de uso previamente elaborados.

5.3 CONECTE IDEIAS

O Conecte Ideias é uma plataforma que permite a construção colaborativa de novas ideias e soluções. Para realizar o estudo foram desenvolvidas duas aplicações cliente para o Conecte Ideias: uma aplicação desenvolvida utilizando os recursos fornecidos pelo ferramental do CReAMA Tools e uma aplicação construída sem o auxílio de CReAMA (o desenvolvedor utilizou algumas *libraries* disponíveis na *Web*). O objetivo foi realizar uma comparação de dificuldade de desenvolvimento de cada versão para ilustrar as vantagens e desvantagens da utilização de CReAMA.

O trabalho foi realizado por dois desenvolvedores diferentes, cada um desenvolvendo o mesmo sistema, mas com recursos diferentes. Vale ressaltar que os requisitos definidos e apresentados abaixo foram os únicos utilizados para a comparação que foi realizada. O sistema móvel Conecte Ideias que não utiliza CReAMA faz parte de um projeto maior e por isso seus requisitos não são completamente mostrados abaixo e o sistema ainda está em desenvolvimento, ou seja, os seus requisitos não foram totalmente definidos:

- Execução do login para acessar os dados da plataforma;
- Visualização de uma lista com todas as ideias da plataforma;
- Visualização de uma lista com as ideias mais recentes;
- Visualização dos dados de ideias específicas, contendo: descrição das ideias, lista de envolvidos e lista de comentários;
- Visualização dos dados de um usuário específico, contendo: lista das ideias que o usuário participa, e lista de atividades recentes;
- Busca de ideias a partir de *input* de usuários;

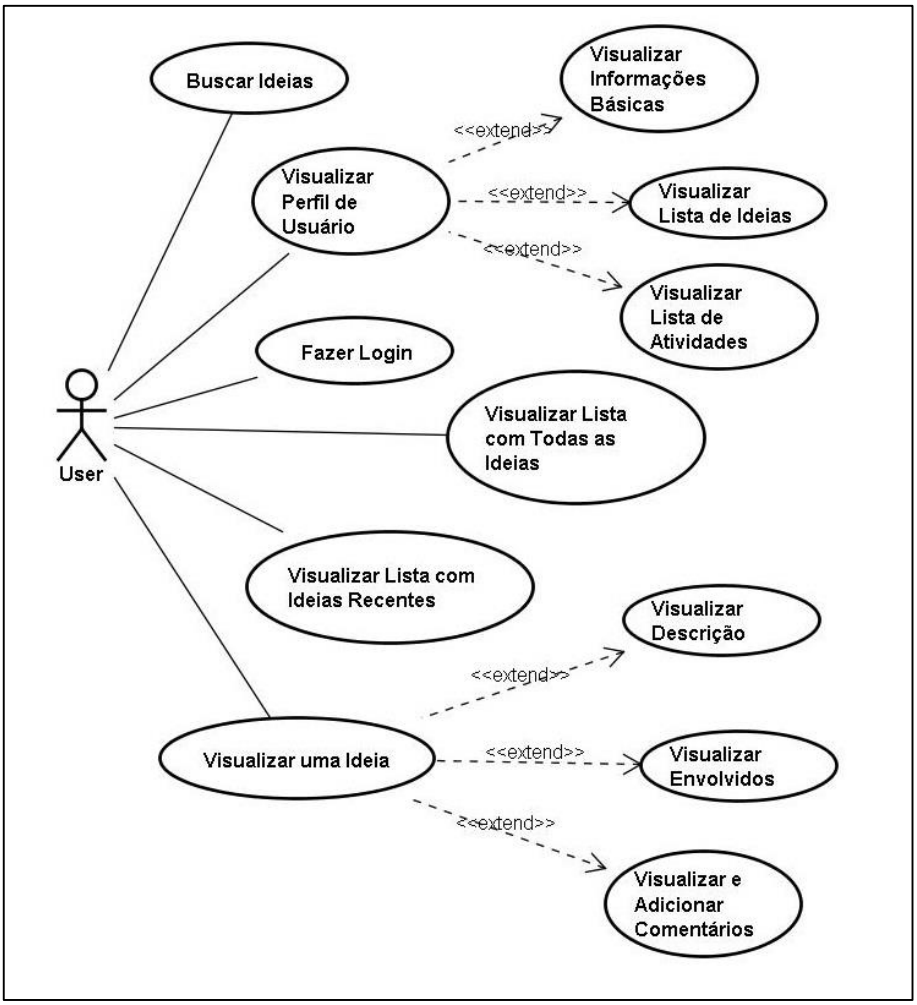


FIGURA 1914 - CASOS DE USO CONECTE IDEIAS

O sistema colaborativo móvel Conecte Ideias construído utilizando CReAMA Tools é composto por dois projetos: o projeto da aplicação e o projeto dos componentes. O projeto da aplicação consiste basicamente de classes que representam a interface gráfica com o usuário (*CRActivity*) e esse projeto utiliza as classes do projeto de componentes para montar a aplicação. A aplicação é montada utilizando essas classes do projeto de componentes que representam componentes gráficos e podem ser acoplados e desacoplados a uma *CRActivity*. A arquitetura do sistema que utilizou CReAMA e sua relação com o CReAMA Tools é apresentado abaixo na figura abaixo:

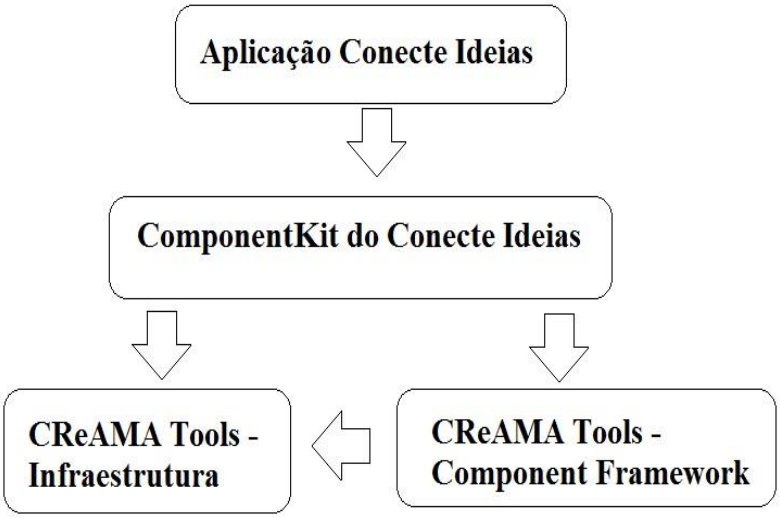
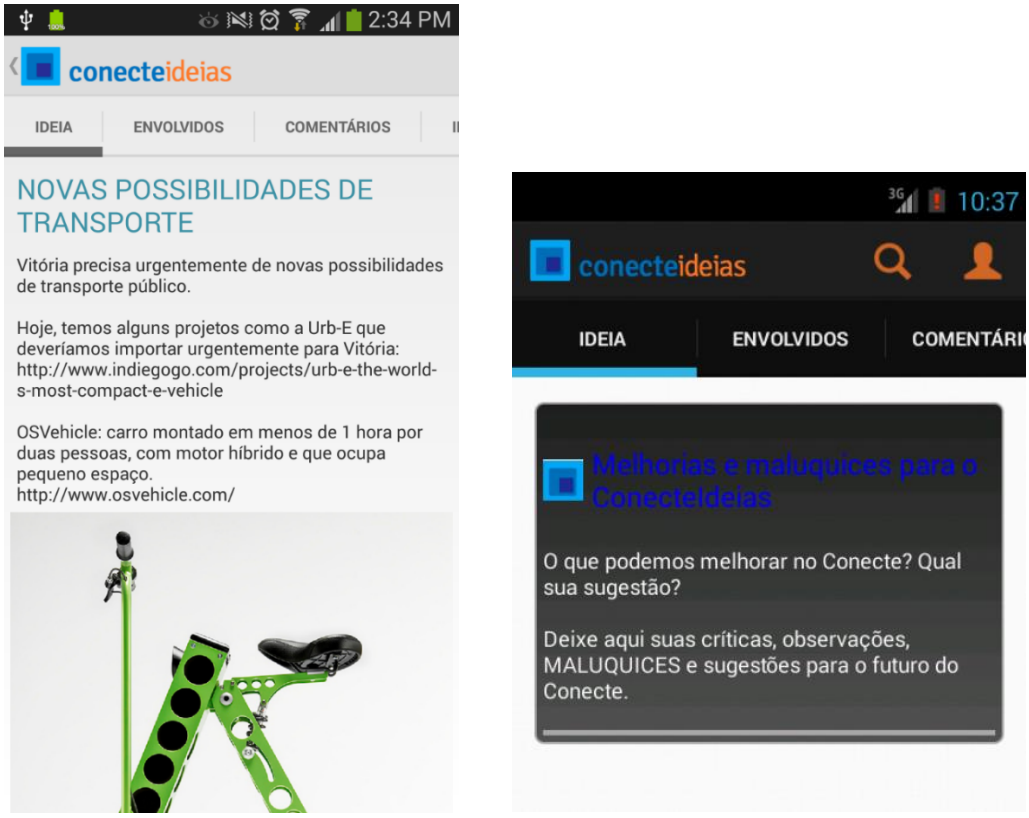


FIGURA 150 - CONECTE IDEIAS E CREAMA

O projeto de componentes faz referência à CReAMA Tools e o projeto da aplicação faz referência ao projeto de componentes. Vale a pena mencionar que o sistema colaborativo móvel Conecte Ideias (CI) construído sem auxílio da arquitetura de referência é composto apenas por um projeto. Abaixo são mostrados *screenshots* dos dois sistemas móveis desenvolvidos:



(A) (B)

FIGURA 21: (A) CONECTE IDEIAS COM CREAMA; (B) CONECTE IDEIAS SEM CREAMA

Para avaliar a eficiência da arquitetura de referência, foi realizada a comparação das duas aplicações com base em métricas de software e levando em conta características específicas da mobilidade. As métricas foram obtidas utilizando o *Eclipse Metrics Plugin* (2014), que aplica as métricas no código e projeto deixando disponível as informações por meio do log de *warnings* do Eclipse IDE.

As métricas utilizadas neste trabalho têm relação direta com estimativa de esforço de desenvolvimento de softwares, sendo elas*: lines of code* (LOC), *number of classes* (NOC) e *weighted methods per class* (WMC) – linhas de código, número de classes e métodos ponderados por classe.

Segundo Yun (2005), das métricas relacionadas a tamanho, a contagem de linhas de código fonte (LOC) é a mais aceita, devido a:

• Facilidade de definir e discutir;

• Facilidade de medir objetivamente;

• É conceitualmente familiar a desenvolvedores de software.

As outras duas métricas, NOC e WMC são utilizadas pelo Modelo de Ponto de Função Orientado a Objetos (*Object Oriented Function Point* - OOFP). Esse modelo foi apresentado em Antioniol et al. (2003) e tem como objetivo estimar o esforço de desenvolvimento em um projeto de software utilizando atributos de tamanho. De acordo com Rosenberg e Hyatt (1997), métricas de tamanho afetam os seguintes atributos de qualidade de software: usabilidade, manutenibilidade, e reusabilidade. Vale mencionar que essas características de qualidade são muito importantes para o presente trabalho por se tratar de uma arquitetura para orientar o desenvolvimento de sistemas orientados a componentes.

Para finalizar o estudo de caso, foi realizada uma análise das características de suporte à mobilidade em cada um dos sistemas móveis implementados, para comparar as duas soluções. Os resultados da avaliação incluindo essa comparação são mostrados a seguir.

5.4 RESULTADOS

Para realizar a avaliação da arquitetura de referência, é preciso antes fazer uma análise da implementação da arquitetura com base nos requisitos definidos anteriormente para verificar de que maneira cada um dos requisitos são atendidos.
Para cada requisito foi feito o questionamento: a implementação da arquitetura atende ao requisito ‘x’? Como todos os requisitos foram atendidos pelo menos parcialmente, as possíveis respostas foram: atende diretamente (AD), provê meios de atender (PMA) e atende parcialmente (AP). Essa análise foi realizada com um *checklist* e o resultado é apresentado na tabela abaixo:

TABELA 2 - VERIFICAÇÃO DOS REQUISITOS

Requisitos	AD	PMA	AP
Ai		X	
Aii	X		
Bi		X	
Bii		X	
Biii	X		
Ci		X	
Cii		X	
Ciii		X	
Di		X	
Dii		X	
Diii		X	
Div		X	
Ei			X
Eii			X
Fi		X	
Fii		X	
Fiii	X		
Gi	X		
Gii		X	
Hi	X		
Hii	X		
Ii	X		

Iii	X		
Iiii			X
Ji	X		
Jii		X	
Jiii	X		
Jiv		X	
Jv		X	
Jvi		X	

É preciso mencionar que muitos requisitos são dependentes do tipo de aplicação. Requisitos de Comunicação (requisitos *D.i* a *D.iv*), por exemplo, devem ser implementados por meio de componentes e não na infraestrutura. O importante neste caso é que a arquitetura possa prover meios de se obter as informações necessárias para que os componentes de domínio da aplicação atendam aos requisitos. Outros exemplos de requisitos que se encaixam nesse caso como foi citado anteriormente no caso de “Awareness” (requisitos *F.i* e *F.ii*) e “Autonomia de Aplicação” (requisitos *B.i* e *B.ii*).

Depois dessa análise foi realizada a aplicação de métricas nos três projetos: projeto do Conecte Ideias baseado em componentes, projeto dos componentes do Conecte Ideias e o Projeto Conecte Ideias sem utilizar CReAMA. Foi realizada a soma dos dois projetos que utilizaram CReAMA e apresentado na tabela como Projeto Conecte Ideias – Com CReAMA. Os resultados são apresentados a seguir:

TABELA 3 - APLICAÇÃO DAS MÉTRICAS NO PROJETO

	LOC	WMC	NOC
Projeto da Aplicação Baseada em componentes	137	47	4
Projeto dos Componentes	1183	380	24
Projeto Conecte Ideias – Com CReAMA	1320	417	28
Projeto Conecte Ideias – Sem CReAMA	1896	701	50

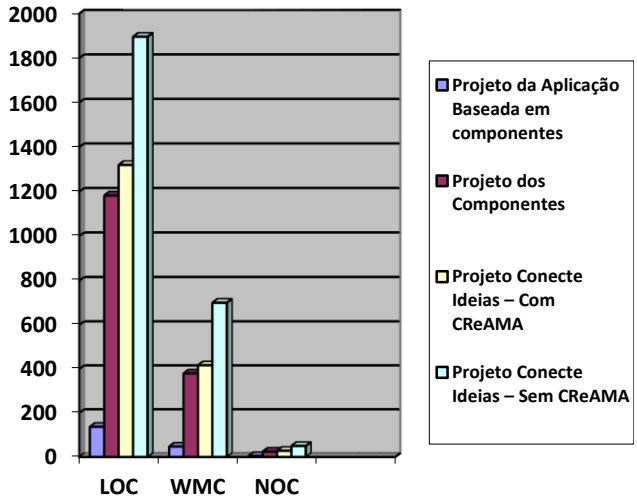


FIGURA 162- GRÁFICO COMPARATIVO DAS MÉTRICAS

Como pode ser observado, CReAMA apresentou vantagem no que diz respeito a todas as métricas. Utilizando CReAMA, a quantidade de linhas de código e classes produzidas do sistema foi muito inferior em relação ao projeto que não utilizou a arquitetura. Foram produzidas 44% menos classes e 31% menos linhas de código. Além disso, a métrica WMC apresentou um resultado 41% menor no sistema que utilizou CReAMA. É importante mencionar que de acordo com Rosenberg e Hyatt (1997), quanto maior o WMC da classe, mais tempo e esforço é necessário para desenvolver e mantê-la.

Além disso, é importante mencionar que o projeto da aplicação do Conecte Ideias utilizando CReAMA apresentou um resultado muito importante para o trabalho. Com o número de linhas de código, número de classes e métodos ponderados por classe muito baixos, foi atingido o requisito de pouco esforço para manter o projeto. Além disso, esse resultado representa também a fácil prototipação, no qual com poucas linhas de código, o montador da aplicação consegue substituir partes de uma *CRActivity* e trocar facilmente um componente por outro.

6 CONCLUSÃO

6 CONCLUSÃO

Neste trabalho foi desenvolvida uma arquitetura de referência, denominada CReAMA e realizada uma avaliação dessa arquitetura que foi criada para orientar o desenvolvimento de sistemas colaborativos móveis orientados a componentes e foi possível observar com uma visão mais abrangente a validade da proposta. Como estudo de caso, dois clientes móveis da plataforma Conecte Ideias foram implementados, um utilizando CReAMA e o outro não. Além disso, foram desenvolvidos dois protótipos no contexto do Groupware-Workbench: Arquigrafia Mobile e o Pergunta-Reposta SyS.Para montar as aplicações, *ToolKits* (ou *ComponentKits*) foram desenvolvidos baseados nessa arquitetura, ou seja, foram desenvolvidos conjuntos de componentes que agregam diversas funcionalidades para apoiar o desenvolvedor. O GW-Android e o IdeaToolKit são ferramentais que oferecem suporte à criação de aplicativos móveis que permitem integrar vários componentes de colaboração.

No caso do GW-Android, o desenvolvedor utiliza apenas os componentes estendidos da plataforma GW, e esses componentes utilizam a infraestrutura que é responsável por lidar com questões técnicas como protocolos e conexões, significando que as utilizar a arquitetura de referência, o desenvolvedor ou montador da aplicação não precisa se preocupar em codificar essas funcionalidades.

No caso do Conecte Ideias, o estudo de caso apresentou resultados favoráveis à CReAMA, permitindo que o usuário do ferramental apenas monte as aplicações com os componentes previamente definidos, atingindo um nível de componentização e agrupamento de componentes satisfatórios, devido à quantidade pequena em todas as métricas, o que implica em que utilizar a arquitetura de referência provê menor esforço para manter o projeto e uma manutenibilidade mais fácil de ser executada.

6.1 TRABALHOS FUTUROS

Como principal trabalho futuro será preciso acertar algumas funcionalidades da infraestrutura, que deixaram um pouco a desejar, devido à tentativa de abordar os muitos requisitos definidos no Capítulo 3. Por exemplo, o componente *Tracker,* devido à mudança de APIs da Google não foi possível utilizá-lo para demonstração, mesmo ele sendo de muita importância pois representa um componente novo que não existia no Groupware-Workbench.

Vale a pena lembrar que o desenvolvimento computacional, que inclui a infraestrutura, os componentes e os protótipos seguindo o padrão para se adequarem aos requisitos não foi uma tarefa fácil e não foi atingida completamente na sua instanciação, ou seja, CReAMA Tools ainda não está completa. Os componentes foram se adequando a CReAMA e a própria arquitetura de referência foi mudando durante todo o desenvolvimento, fazendo com que muito esforço fosse preciso para manter os componentes. Assim, outro trabalho futuro vai ser no desenvolvimento da instância computacional, CReAMA Tools, que ainda não atende totalmente à todos os requisitos definidos na arquitetura de referência.

6.2 DISCUSSÃO

Como pode ser observado no Capítulo 2, sistemas colaborativos móveis baseados em componentes apresentam uma grande dificuldade de implementação levando em contas diversos aspectos. Juntamente a esse fato tem a questão do próprio desenvolvimento ser colaborativo entre os desenvolvedores do GW (USP) e do GW-Android (UFES), com muita troca de informação e às vezes alguns atrasos no que diz respeito à resposta de dúvida ou questões relevante para o desenvolvimento.

É importante mencionar que a arquitetura de referência visa abordar os 32 requisitos levantados na fase de definição da arquitetura, o que tornou a implementação de CReAMA Tools por si só um trabalho complexo e além de CReAMA Tools, foram desenvolvidos dois *ToolKits* e quatro protótipos de sistemas colaborativos móveis, o que exigiu um esforço considerável de implementação dos

protótipos computacionais, mesmo que estes não tenham atendido completamente aos requisitos.

É possível concluir que ferramentas para apoiar desenvolvedores de software são muito importantes de uma forma geral, mas no que diz respeito à colaboração móvel, ferramentas se tornam fundamentais principalmente por se tratar de uma área multidisciplinar que lida com muitas questões técnicas. Porém, sem uma base de requisitos ou modelos que mapeiam esses requisitos para se apoiar, muitas vezes o ferramental pode se tornar inutilizado ou pouco útil.

REFERÊNCIAS

Antoniol, G.; Fiutem, R.; Lokan, C. "Object-Oriented Function Points: An Empirical Validation", Empirical Software Engineering, vol. 8, no. 3, pp.225 -254. 2003.

Alaya, M. B.; Baudin, V.; Drira, K. Dynamic deployment of collaborative components in service-oriented architectures. In: 11th International Conference of New Technologies in Distributed Systems (IEEE NOTERE, 2011).

Bendel, S.; Schuster, D. “WatchMyPhone - Providing developer support for shared user interface objects in collaborative mobile applications”, in Proc. IEEE PERCOM Workshops, 2012.

Berkenbrock, C. D. M. Uma estratégia para garantir coerência de cache e percepção em sistemas cooperativos com apoio à mobilidade. Tese de Doutorado – Instituto Tecnológico de Aeronáutica. São Paulo, 2009.

Berkenbrock, C. D. M.; Hirata, C. M.; Fernandes, C. T.; Pichiliani, M. C. Requisitos de Usabilidade para o Desenvolvimento e Avaliação de Aplicações Cooperativas Móveis. VI Simpósio Brasileiro de Sistemas Colaborativos. 2009.

Berkenbrock, C. D. M.; Da Silva, A. P. C.; Hirata, C. M. Designing and Evaluating Interfaces for Mobile Groupware Systems. Proceedings of the 2009 13th International Conference on Computer Supported Cooperative Work in Design. 2009.

Borghoff, U. M.; Schilichter, J. H. Computer-Supported Cooperative Work: Introduction to Distributed Applications. Munique, Alemanha: Springer, 2000.

Buschmann, F., Henney K., Schmidt, D. C. Pattern Oriented Software Architecture, Volume 5: On Patterns and Pattern Languages. Wiley & Sons, 2007.

Byrne, P. “MUSE - Platform For Mobile Computer Supported Collaborative Learning,” Ph.D. dissertation, University of Dublin, 2011.

Eclipse Metrics Plugin. Disponível em http://eclipse-metrics.sourceforge.net/. Acesso em Maio de 2014.

Ellis, C.A., Gibbs, S.J. & Rein, G.L. (1991): Groupware - Some Issues and Experiences. Communications of the ACM, Vol. 34, No. 1, pp. 38-58.

Filippo, D., Filho, J. V., Endler, M., Fuks, H. Mobilidade e ubiqüidade para colaboração. In: Sistemas Colaborativos. Rio de Janeiro: Elsevier, 2011. 294-315.

Fonseca, B.; Paredes, H.; Sousa, J. P.; Martins, F. M.; Carrapatoso, E. SAGA Reloaded: towards a generic platform for developing cooperative applications. 13th International Conference on Computer Supported Cooperative Work in Design (CSCWD 2009). Santiago, Chile.

Goadrich, M. H. and Rogers, M.P. Smart smartphone development: iOS versus android. Proceedings of the 42nd ACM technical symposium on Computer science education. Pages 607-612. 2011.

Gerosa, M. A. Desenvolvimento de Groupware Componentizado com Base no Modelo 3C de Colaboração. Tese de Doutorado - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática. Rio de Janeiro, 2006.

Gerosa, M.A.; Fuks, H. A. Component Based Workbench for Groupware Prototyping. 1st Workshop on Software Reuse Efforts (WSRE), 2nd Rise Summer School, 27-28 de outubro de 2008, Recife.

Gomes, R. L, Willrich, R., Rivera, G. D. H. Arquiteturas distribuídas para sistemas colaborativos. In: Sistemas Colaborativos. Rio de Janeiro: Elsevier, 2011. pp. 328-346.

Groupware Workbench. Groupware Workbench: Descrição do Projeto. Disponível em: http:/www.groupwareworkbench.org.br . Acesso em Maio de 2013.

Herskovic, V., Ochoa, S.F., Pino, J.A., Neyem, A. “General Requirements to Design Mobile Shared Workspaces”. Proceedings of CSCWD 2008. IEEE Press. Xi’an, China. April 16-18, 2008, 582-587.

Herskovic, V., Ochoa, S.F., Pino, J.A. Modeling Groupware for Mobile Collaborative Work. IEEE Press, Los Alamitos, CA. 13th International Conference on Computer Supported Cooperative Work in Design (CSCWD'09), Santiago, Chile, April 22-24, 2009, 384-389.

Herskovic, V.; Ochoa, S. F.; Pino, J. A.; Neyem, A. The Iceberg Effect: Behind the User Interface of Mobile Collaborative Systems. Journal of Universal Computer Science, vol. 17, no. 2 (2011), 2011,183-202.

Le, H. N.; Nygard, M. A transaction model for Supporting mobile Collaborative Works. In Proc. CTS’07: 7th International Symposium on Collaborative Technologies and Systems, 2007.

Lukosch, S. “Seamless Transition between Connected and Disconnected Collaborative Interaction”; J.UCS (Journal of Universal Comp. Science), 14, 1 (2008), 59-87.

Mamani, E.Z.S., Gerosa, M.A. Cálculo de Reputação em Redes Sociais. VIII Simpósio Brasileiro de Sistemas Colaborativos - SBSC 2011, Paraty - RJ.

Melotti, M.; Gomes, R. L. Extensão do Groupware Workbench para Prototipação de Sistemas Colaborativos Móveis. In: IX Brazilian Symposium in Collaborative Systems. Workshop de Teses e Dissertações. 2012.

Melotti, M.; Gomes, R. L.; Nunes, V. R. O.; Gerosa, M. A. CReAMA: A Component-Based Reference Architecture for Collaborative Mobile Applications. Proceedings of the X Brazilian Symposium in Collaborative Systems. 2013.

Michalsky, S. Componentes de Software no desenvolvimento de aplicações colaborativas para Web: Evolução da plataforma Groupware Workbench. Dissertação de Mestrado – Universidade de São Paulo, Departamento de Matemática e Estátisca. São Paulo, 2012.

Neyem, A., Ochoa, S.F., Pino, J.A. “Integrating Service-Oriented Mobile Units to Support Collaboration in Ad-hoc Scenarios”; J.UCS (Journal of Universal Computer Science), 14, 1 (2008), 88-122.

Ochoa, S.F., Neyem, A., Pino, J., Borges, M. “Supporting Group Decision Making and Coordination in Urban Disasters Relief Efforts”; Journal of Decision Systems 16, 2 (2007), 143-172.

Oliveira, L. Santos de. Funcionalidades colaborativas no compartilhamento de conteúdo em redes sociais na Web 2.0: Uma engenharia de domínio baseada no modelo 3C de colaboração. Dissertação de Mestrado – Universidade de São Paulo, Departamento de Matemática e Estátisca. São Paulo, 2010.

Pinelle, D., Gutwin, C. “Loose Coupling and Healthcare Organizations: Deployment Strategies for Groupware”; Computer Supported Cooperative Work Journal, 15, 5-6 (2006), 537-572.

Roseman, M. and Greenberg, S. (1992). GroupKit: A Groupware Toolkit. Conference Companion - ACM SIGCHI Conference on Human Factors in Computing Systems.

Rosenberg, L. H.; Hyatt, L. Software Quality Metrics for Object- Oriented Environments. Crosstalk Journal. 1997.

Richardson, L.; Ruby, S. RESTful Web Services: Web Services for theReal World. O’Reilly Media Inc, 2007.

Tsirulnik, G. Mobile collaboration for increase employee productivity: Study, Mobile Marketer, 2009.

Tse, E. and Greenberg, S. Rapidly prototyping Single Display Groupware through the SDGToolkit. Proceedings of the fifth conference on Australasian user interface - Volume 28 Pages 101-110. 2004.

Truong, H-L, Dustdar, S. Service-Oriented Architecture for Mobile Services. In *Handbook of Mobile Systems Applications and Services*, Kumar, A. and Xie, B. *editors*. Auerbach Publications, 2012, 612 p.

VISION MOBILE. Developer Economics 2013-Developer Tools-the Foundation of the App Economy. Disponível em: http://www.visionmobile.com/product/developer-economics-2013-the-tools-report. Acesso em maio de 2013.

Wulf, V.; Pipek, V.; Won, M. Component-based tailorability: Enabling highly flexible software applications. Int. J. Hum.-Compt. Stud. 66(1), 2008.

Yun, S. J. Productivity Prediction Model Based on Bayesian Analysis and Productivity Console. Dissertation, Texas A&M University, 2005.